THE MEASUREMENT OF FLOW VELOCITY DISTRIBUTION

by

MICHAEL DANN

Submitted to the University of Cape Town in part fulfilment of the requirements for the degree of Master of Science in Engineering.

September 1981.

DECLARATION

I wish to certify that this thesis, unless indicated to the contrary in the text, is my own work.

Signed by candidate

ABSTRACT

A method for the improvement of the range and accuracy achieved by the crosscorrelation flowmeter is investigated. The principles of the flowmeter operation and fundamental digital signal processing techniques are reviewed. The process of the Fourier transform deconvolution is investigated. Computer simulation of the flow system is described and is shown to require impractical amounts of computer time to achieve the necessary averaging times. Consequently, correlation and velocity profile measurements are made from an experimental flow rig. A waveform analysis program is used to analyse these measurements. The Fourier transform deconvolution is shown in this case to have poor noise immunity. For this reason, an alternative method of Bayesian deconvolution is investigated. The correlation functions measured from the experimental flow rig are deconvolved using the Bayesian deconvolution algorithm. The resulting transit time distribution is shown to converge to the transit time distribution obtained from the velocity profile measurements. From an analysis of the flow signals the velocity distribution of the flow may thus be found.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Mr. J.R. Greene, for his interest in the project and for the many useful discussions and suggestions.

I would also like to thank Dr. D.E.H. Naudé for initiating this research and his enthusiasm and guidance.

The National Institute for Metallurgy has provided generous financial support for the project.

I am indebted to all the members fo staff of the Department of Electrical Engineering at UCT for their comments and assistance.

Finally, thanks also go to my father, Mr. R.A. Dann, for reading and correcting the draft.

CONTENTS

1.1 INTRODUCTION

In many industries it has become increasingly important to be able to measure accurately the volume flowrate of a variety of fluids. This variety encompasses abrasive solids in suspension to blood flow in arteries.

The ultrasonic crosscorrelation flowmeter has been well developed and is at this stage suitable for flow measurement of industrial slurries. This flowmeter has at present a dynamic range of about 10:1 and an accuracy of 1% once calibrated for the particular velocity profile characterising the flow. The dynamic range over which the flowrate can be measured is limited in that large errors may be introduced if the particular flow velocity for which it has been calibrated were to change. If, for example, the instrument is calibrated for fully developed turbulent flow and the flow changes to fully developed laminar flow (possibly outside the 10:1 measurement range) errors of up to 33% may result (Reference 1.1).

The aim of this project is to investigate a technique to improve the range of operation and the accuracy of the crosscorrelation flowmeter. By determining the velocity

distribution in the flow being measured, some correction may be made to eliminate errors introduced due to variations of the velocity profile. It is proposed that this may be achieved by suitably processing the signals provided by the crosscorrelation flowmeter.

1.2 PRELIMINARY MATHEMATICS

The Fourier transform of a function of time h(t), satisfying the Dirichlet conditions (given in Reference 1.2) is:

$$H(f) = \int_{-\infty}^{\infty} h(t)\, e^{-j2\pi ft}\, dt \qquad \ldots 1.1$$

and the inverse Fourier transform is described by:

$$h(t) = \int_{-\infty}^{\infty} H(f)\, e^{j2\pi ft}\, df \qquad \ldots\ldots 1.2$$

The concept of crosscorrelation of two functions may be described as a measure of the similarity between the functions for different values of delay between them. This is more exactly described by:

$$\phi_{xy}(\tau) = \int_{-\infty}^{\infty} X(\theta)\, y(\tau + \theta)\, d\theta = X(t) \,\theta\, y(t) \qquad \ldots 1.3$$

where $\phi_{xy}(\tau)$ is the crosscorrelation function of x(t) and y(t) and θ represents the crosscorrelation operator.

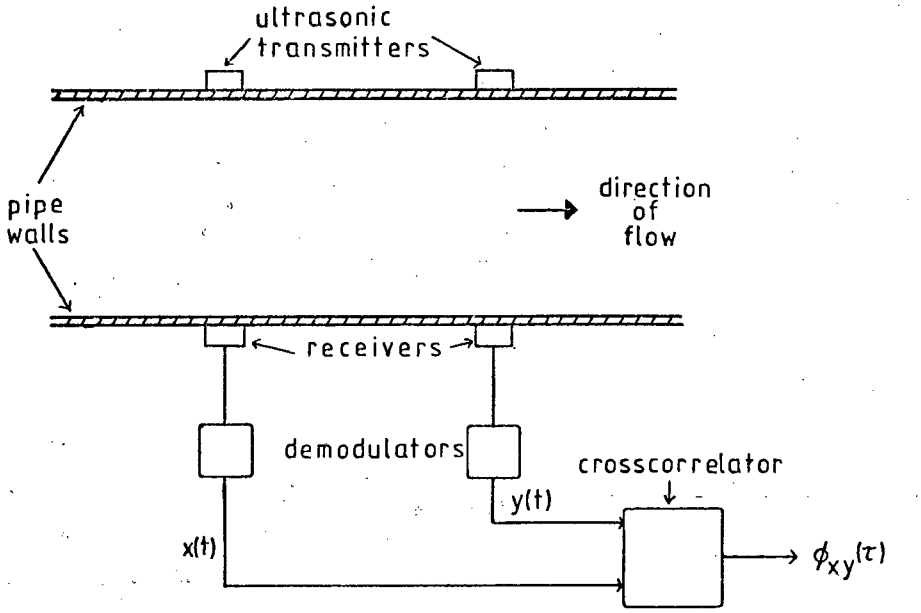


Figure 1.1 General schematic of the crosscorrelation flowmeter.

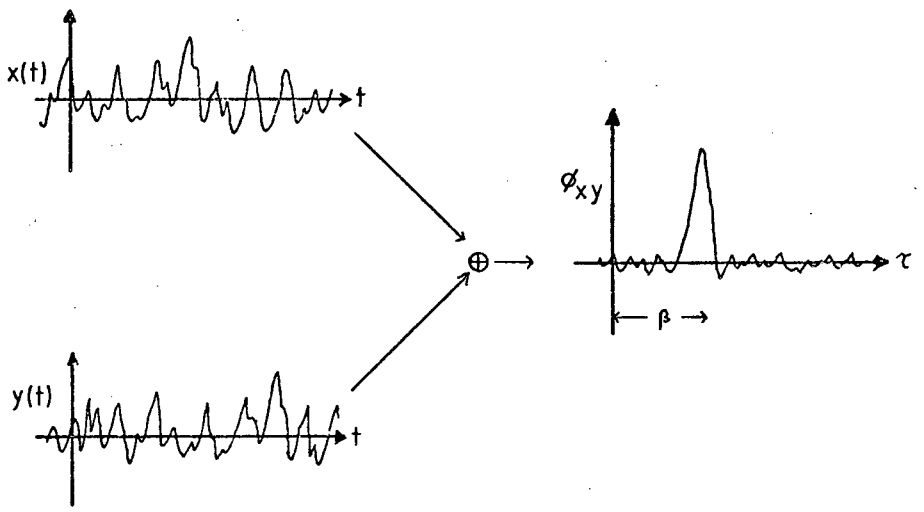


Figure 1.2 Typical flow signals and their crosscorrelation.

1.3 THE ULTRASONIC CROSSCORRELATION FLOWMETER

Ultrasonic crosscorrelation flowmetering is achieved by passing two ultrasonic beams, some distance apart, through the flowing stream. Variations in density and eddies present in the material being transported modulate the amplitude and frequency of the received signal. The flow signals, one from the upstream and the other from the downstream transducer, may be derived from either variations in amplitude or variations in frequency of the received signals. Variations in density and eddies at the upstream transducer appear at the downstream transducer a short while later having undergone some modifications in transit between the transducers. The eddies, however, tend to have a finite lifetime and their modulation effect on the ultrasonic beam changes exponentially with time. The positions of the eddies and the variations in their relative density is modified by the velocity profile of the flow. If the crosscorrelation of the two flow signals is calculated then the resulting function $\emptyset_{xy}(\tau)$ will exhibit a global maximum at the delay at which the two signals are separated. The velocity of the material being transported may be found since the delay of β seconds is a measure of the time taken for this material or fluid to travel the distance between the two transducers and the velocity is inversely proportional to this delay.

1.4 THE FLOW SYSTEM

Consider a pipe as shown in Figure 1.1 filled with a homogeneous, stationary fluid. The outputs of the two demodulators will then be zero. Now a particle of higher density than the fluid in the pipe, travelling with a velocity V_o, in the centre of the pipe, passes through the upstream ultrasonic beam. This causes a change in the upstream flow signal. The same change will appear in the downstream flow signal as the particle passes through the downstream ultrasonic beam. Crosscorrelating these two flow signals will produce a maximum in the crosscorrelation function at the delay corresponding to the time taken for the particle to travel from one beam to the other. (When considering the flow of an eddy a modification of the eddy occurs while travelling between the ultrasonic beams due to the eddy decay. This has the effect of reducing the height of the crosscorrelation function as the distance between the transducers is increased.) Assume now that when the aforementioned particle enters the upstream beam, another particle travelling at a lower velocity V_i also crosses the upstream beam. They will cross the downstream beam at different times and the resulting crosscorrelation of the two flow signals will exhibit two peaks, one corresponding to velocity V_o and the other to velocity V_i.

Instead of particles, consider now a flow channel divided

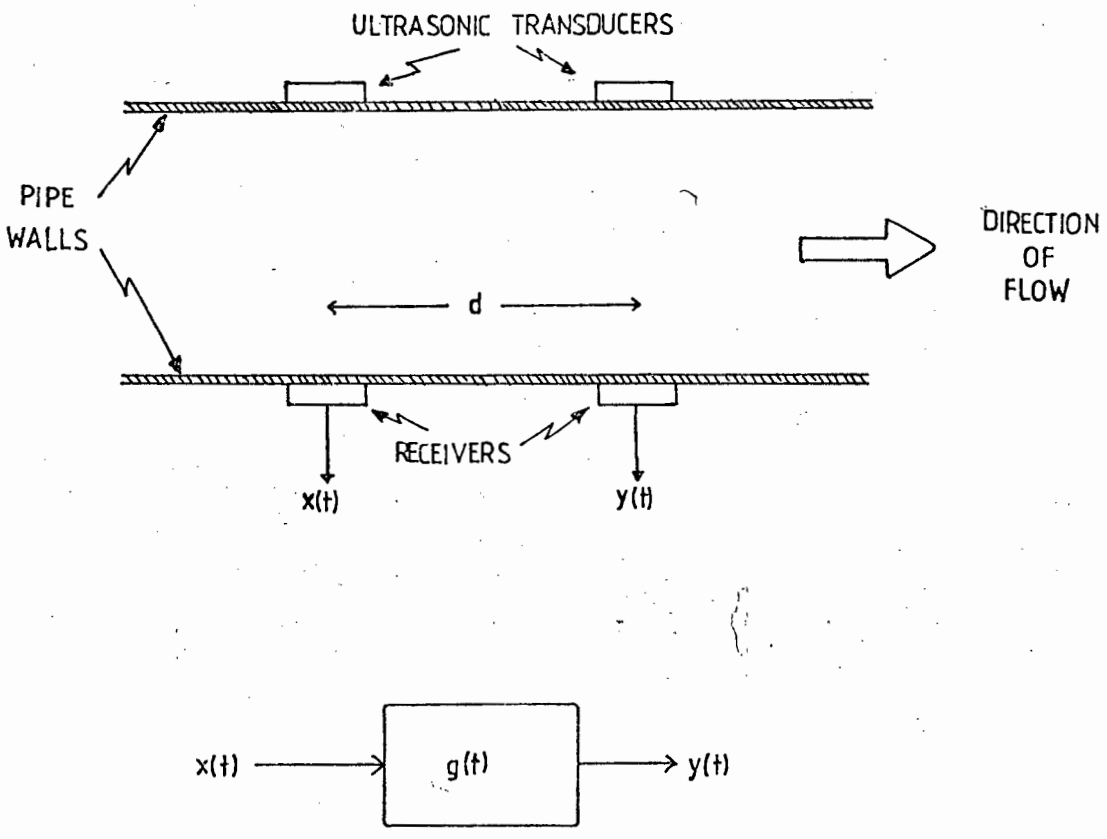


Figure 1.4 The flow system.

by a partition into two subchannels. If the fluid in the two subchannels is travelling at V_1 and V_2, then the crosscorrelation of the upstream and downstream flow signals will exhibit two peaks at delays corresponding to the velocities V_1 and V_2. The relative heights of the two peaks will depend on the signal strengths due to each subchannel. The signal strengths are directly related to the respective subchannel widths. More simply - the wider the subchannel with fluid at velocity V_i, the higher the peak of the crosscorrelation function corresponding to this velocity. Now a parabolic velocity profile representing laminar flow can be represented by increasing the number of subchannels. The individual peaks of the resulting crosscorrelation will then merge to form a composite "smeared" crosscorrelation function the shape of which must contain information about the velocity profile.

This leads on to the idea of the flow between the two transducers as a system with the input being the upstream flow signal and the output the downstream flow signal. The impulse response of the system may be interpreted as the transit time distribution (Reference 1.3). Thus:

$$w(t) = g(t)$$

where w(t) is the transit time distribution. The determination of the system impulse response is thus a system identification problem.

If the system is linear then the output y(t) is the convolution of the input x(t) and the system impulse response g(t).

$$y(t) = \int_{-\infty}^{\infty} x(\theta)\, g(t - \theta)\, d\theta \qquad \ldots 1.4$$

or

$$y(t) = x(t) * g(t)$$

where * represents the convolution operator.

Various methods exist for finding the impulse response of a system, the simplest being to excite the system with a deterministic signal such as a step, ramp or impulse and observe the output. Random signals which may be applied or occur naturally may also be used to excite the system. Then the output of the system may be described by:

$$\phi_{xy}(\tau) = \int_{-\infty}^{\infty} \phi_{xx}(\theta)\, g(\tau - \theta)\, d\theta$$

where $\phi_{xy}(\tau)$ is the crosscorrelation of the upstream and downstream flow signals

$\phi_{xx}(\tau)$ is the autocorrelation of the upstream flow signal

$g(t)$ is the system impulse response.

(This expression may be derived by Fourier transforming both sides of equation 1.4 above and assumes the signals to be both stationary and ergodic).

The velocity distribution may be derived from the transit time distribution:

$$W(v) = \frac{-w(t)}{dv/dt} = w(t)\ \frac{t^2}{L} \qquad \ldots 1.5$$

Thus the correlation measurements should yield not only the mean velocity but also the complete velocity distribution independent of the transducer properties.

1.5 THE PROJECT IN PERSPECTIVE

The field of ultrasonic crosscorrelation flowmetering has been fairly well researched (References 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 1.10, 1.11) and of particular interest is the work done by Leitner (Reference 1.12). He has developed a crosscorrelation flowmeter using a microprocessor controlled two-point difference correlator. The aim of my work is to apply digital signal processing techniques to improve the range and accuracy of this microprocessor controlled correlator. Ultimately one may envisage an ultrasonic flowmeter incorporating a correlator and microcomputer which calculates the auto-correlation and crosscorrelation functions of the flow signals and then deconvolves these two functions to yield the flow system impulse response. This impulse response may then undergo further processing to yield the entire velocity distribution or simply a correction factor for the measured flowrate.

The task at hand is thus to develop and test a method for performing the required deconvolution.

REFERENCES - CHAPTER 1

1.1 Gessner, U.: The Performance of the Ultrasonic Flowmeter in Complex Velocity Profiles. IEEE Transactions on Biomedical Engineering, Vol. BME-16, No. 2, April 1969, pp.139-142.

1.2 Speigel, M.R.: Advanced Calculus. Schaum's Outline Series, New York, McGraw-Hill, 1963, pp.299.

1.3 Mesch, F. et al: Transit Time Correlation - A Survey on Its Applications to Measuring Transport Phenomena. Transactions of the ASME, Journal of Dynamic Systems, Measurement and Control, Vol. 96, No. 4, December 1974, pp.414-420.

1.4 Flemons, R.S.: A New Non-Intrusive Flowmeter. National Bureau of Standards Special Publication 484. Proceedings of the Symposium on Flow in Open Channels and Closed Conduits, Gaithersburg, February 1977, pp.23-25.

1.5 Bazerghi, H. and Serdula, K.J.: Estimation and Reduction of Errors in Flow Measurements Which Use Correlation Techniques. Progress in Nuclear Energy, Vol. 1. G.B. Pergamon Press, 1977, pp.629-648.

1.6 Naude, D.E.H. and Leitner, J.R.: Slurry Flow-Rate Measurement Using Correlation. South African Institute of Measurement and Control, Symposium on Metallurgical Process Instrumentation, Randburg, South Africa, September 1978.

1.7 Mesch, F. and Kipphan, H.: Solids Flow Measurement by Correlation Methods. Opto-Electronics, Vol. 4, June 1972, pp.451-462.

1.8 Coulthard, J.: The Principle of Ultrasonic Crosscorrelation Flowmetering. Measurement and Control, Vol. 8, February 1975, pp.65-70.

1.9 Ong, K.H. and Beck, M.S.: Slurry Flow Velocity, Concentration and Particle Size Measurement Using Flow Noise and Correlation Techniques. Measurement and Control, Vol. 8, November 1975, pp.453-463.

1.10 Le Guennec, B. et al: Analysis of a Horizontal Solid-Liquid Pipe Flow by a Crosscorrelation Method. International Journal of Multiphase Flow, Vol. 4, 1978, pp.511-521.

1.11 Henry, R.M. and Al Chalabi, L.: Microprocessor Applications to Velocity Measurement by Crosscorrelation.
IMEKO Conference on Flow Measurement, Moscow, May 1979.

1.12 Leitner, J.R.: Slurry Flowmetering Using Correlation Techniques.
Ph.D. Thesis, University of Cape Town, 1979.

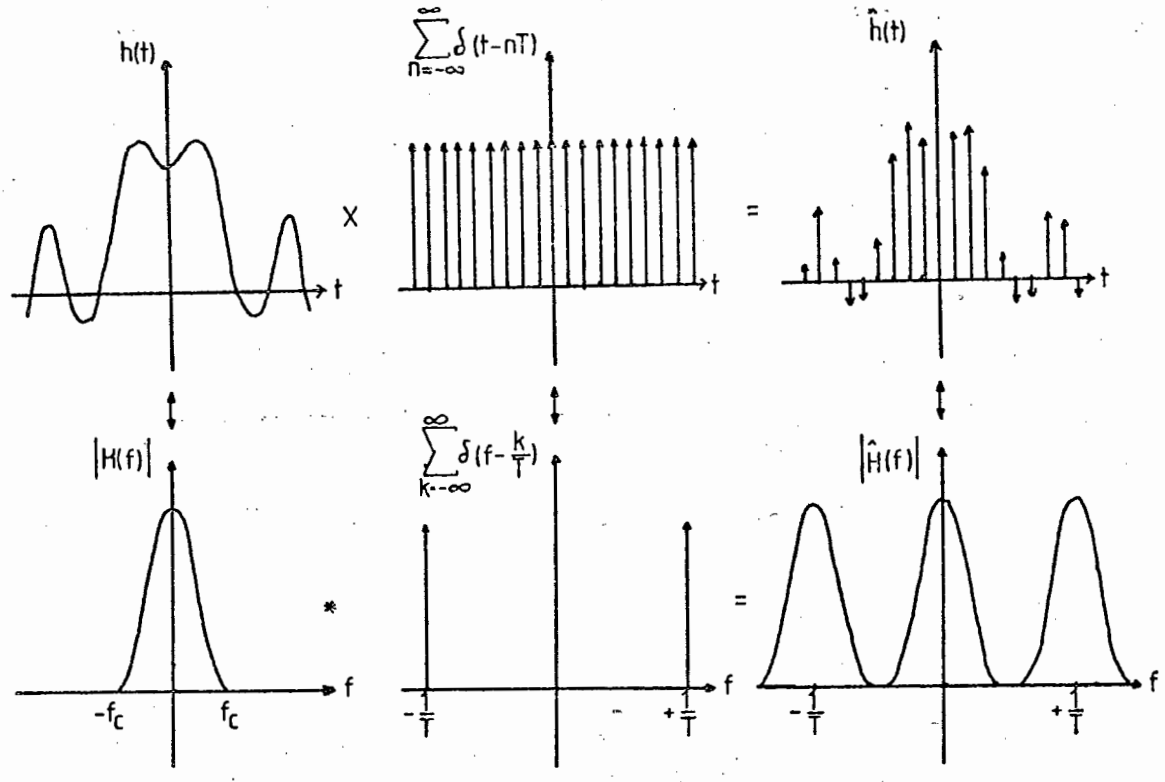


Figure 2.1 The sampling process with $\frac{1}{T} > 2\ f_c$.

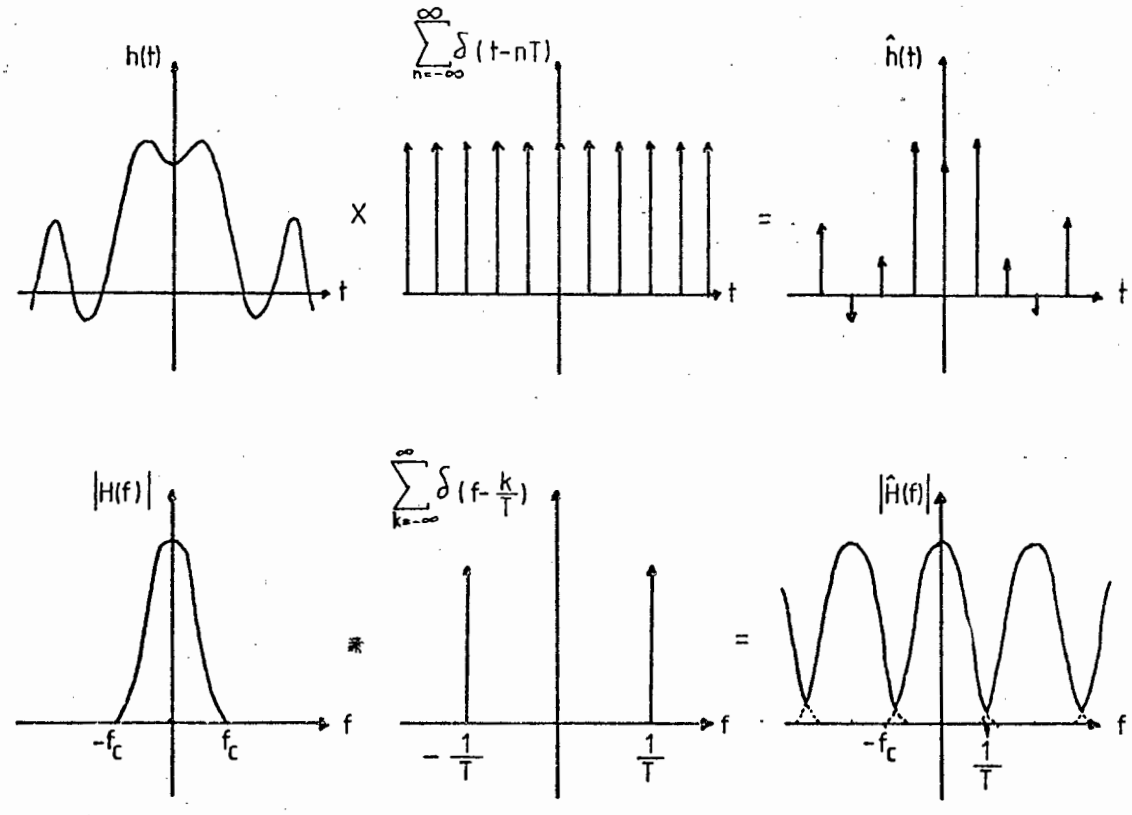


Figure 2.2 The sampling process with $\frac{1}{T} < 2\ f_c$.

DIGITAL SIGNAL PROCESSING

2.1 SAMPLING

Let $\hat{h}(t)$ be a sampled representation of the continuous function h(t). This sampled function $\hat{h}(t)$ may be considered as the continuous function h(t) multiplied by an infinite sequence of unit impulses.

$$\hat{h}(t) = h(t) . \sum_{n=-\infty}^{\infty} \delta (t-nt) = \sum_{n=-\infty}^{\infty} h(t)\, \delta (t-nT) = h(nT)$$

for $n \in I$ and T the sampling interval.

The Fourier transform of $\hat{h}(t)$ may be found by noting that multiplication in the time domain is equivalent to convolution in the frequency domain, thus:

$$\hat{H}(f) = H(f) * \sum_{k=-\infty}^{\infty} \delta \left(f - \frac{k}{T}\right)$$

where H(f) and $\hat{H}(f)$ are the Fourier transforms of h(t) and $\hat{h}(t)$ respectively. These operations are represented graphically in Figure 2.1.

In this diagram we see that $1/T > 2 f_c$ where f_c is the highest frequency component present in the signal h(t). If the sampling interval were increased to a value where sampling took place less often than twice each cycle of

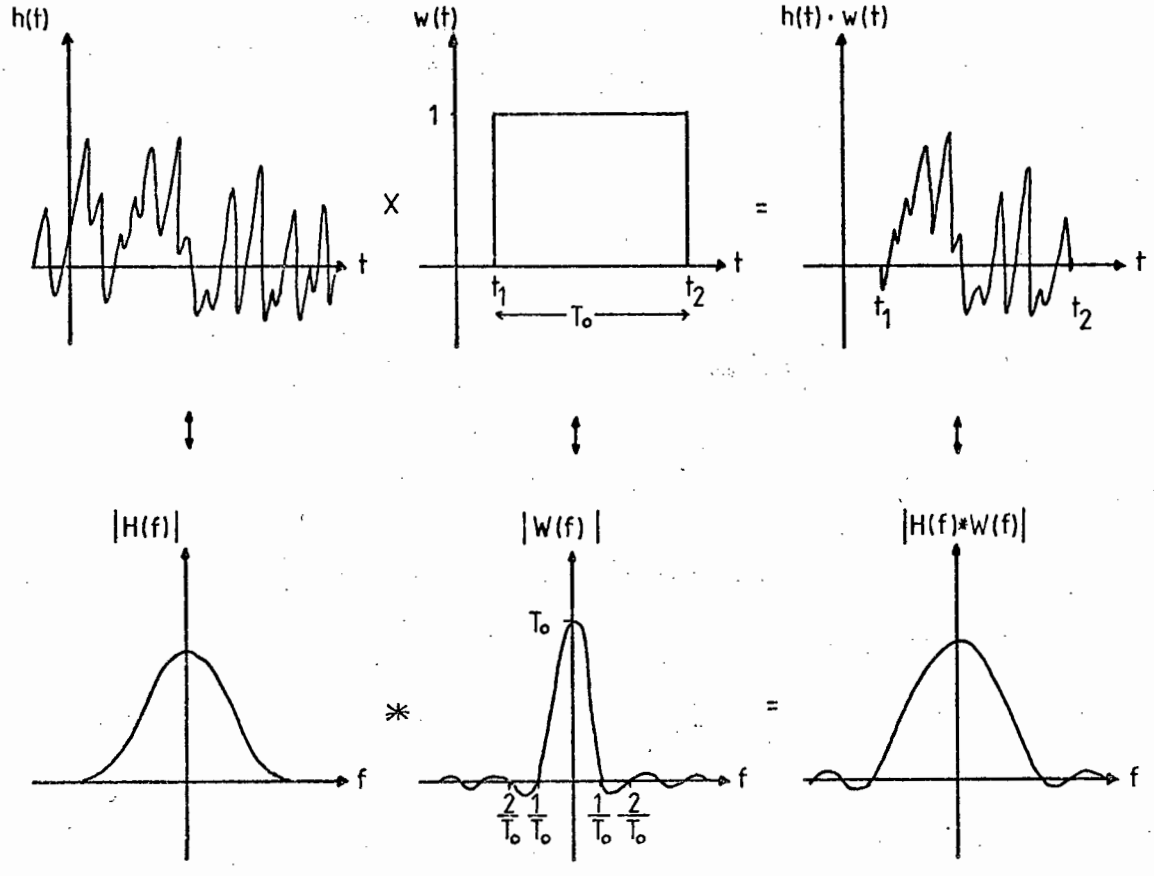


Figure 2.3 Application of a rectangular window function.

the highest frequency component f_c, then the phenomenon known as aliasing would occur. Aliasing implies an overlapping and distortion of the desired Fourier transform of the sampled function. This is shown in Figure 2.2.

To eliminate aliasing one must ensure that $T > 2 f_c$. This is usually achieved by low pass filtering h(t) before sampling at a frequency of anything between three and five times the cut off frequency of the anti-aliasing filter.

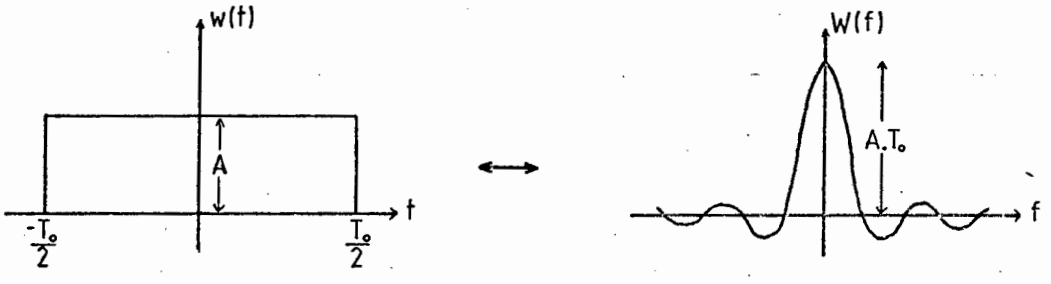
2.2 WINDOWING

Ultimately we wish to apply the relations defined for continuous functions to sampled waveforms using a computer. It is obviously impossible to deal with infinite records of data so a finite length sample of the data must be used. The most obvious method is simply to select N data starting at time t_1 and ending at t_2. This is the same as applying a rectangular window function to the infinite sequence. As multiplication in the time domain is equivalent to convolution in the frequency domain, the resulting frequency domain representation has some ripple added to it as shown in Figure 2.3.

By examining Figure 2.3 one can appreciate the desire to keep the sidelobes of the Fourier transformed window function as small as possible while keeping the width as

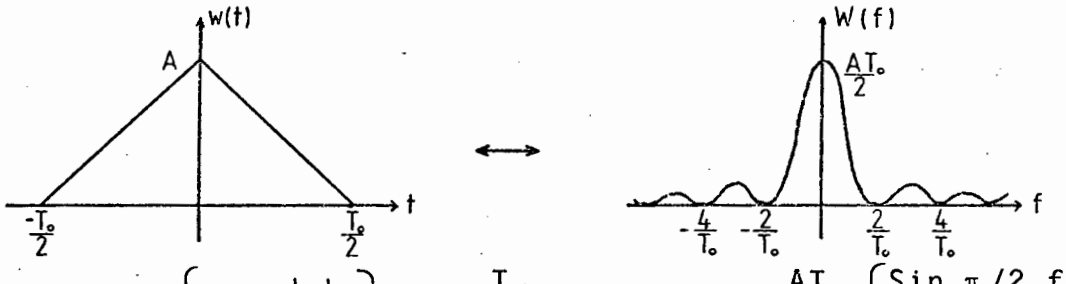
narrow as possible. This can be done by either using many data points, that is, making T_o large, or by a wise choice of the window function. The selection of window functions has been well researched and Childers and Durling (Reference 2.1) refer to this field as that of "window carpentry". Some of the more common window functions and their transforms are given here.

Rectangular Window



$$w(t) = A\left\{u(t+\frac{T_o}{2}) - u(t-\frac{T_o}{2})\right\} \qquad w(f) = AT_o\,\frac{\text{Sin}\ \pi\, fT_o}{\pi\, fT_o}$$

Triangular Window

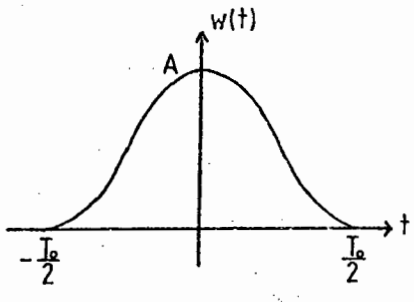


$$w(t) = A\left\{1 - \frac{|t|}{T_o/2}\right\}, \quad |t| \leq \frac{T_o}{2}$$

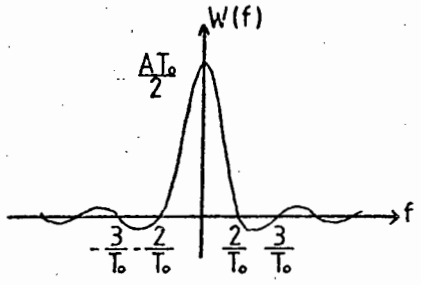
$$= 0 \qquad , \quad |t| > T_o/2$$

$$w(f) = \frac{AT_o}{2}\left\{\frac{\text{Sin}\ \pi/2\ fT_o}{\pi/2\ \ fT_o}\right\}^2$$

Hanning Window (after Von Haan)



$\longleftrightarrow$

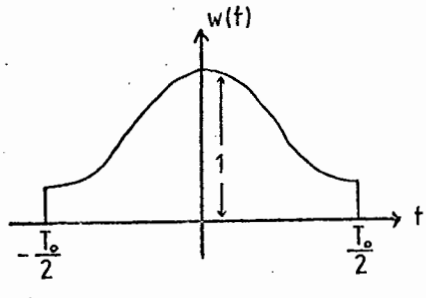


$$w(t) = A \cos^2 \frac{\pi t}{T_o} = A\left\{1+\cos \frac{2\pi t}{T_o}\right\}, \quad t < \frac{T_o}{2}$$

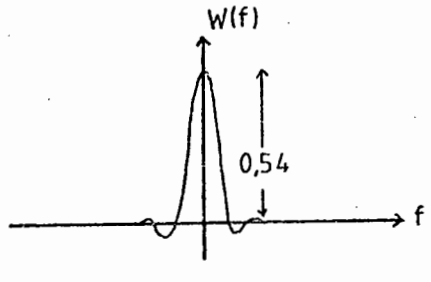
$$= 0 \quad , \quad |t| \geq \frac{T_o}{2}$$

$$w(f) = \frac{AT_o}{2} \cdot \frac{\sin \pi f T_o}{(\pi f T_o)(1 - f^2 T_o^2)}$$

Hamming Window



$\longleftrightarrow$



$$w(t) = 0{,}54+0{,}46 \cos\frac{2\pi t}{T_o} \quad , \quad |t| < \frac{T_o}{2}$$

$$= 0 \quad , \quad |t| \geq \frac{T_o}{2}$$

$$w(f) = \frac{[0{,}54\pi^2 - 0{,}08(\pi f T_o)^2] \sin \pi f T_o}{f T_o \, (\pi^2 - \pi^2 f^2 T_o^2)}$$

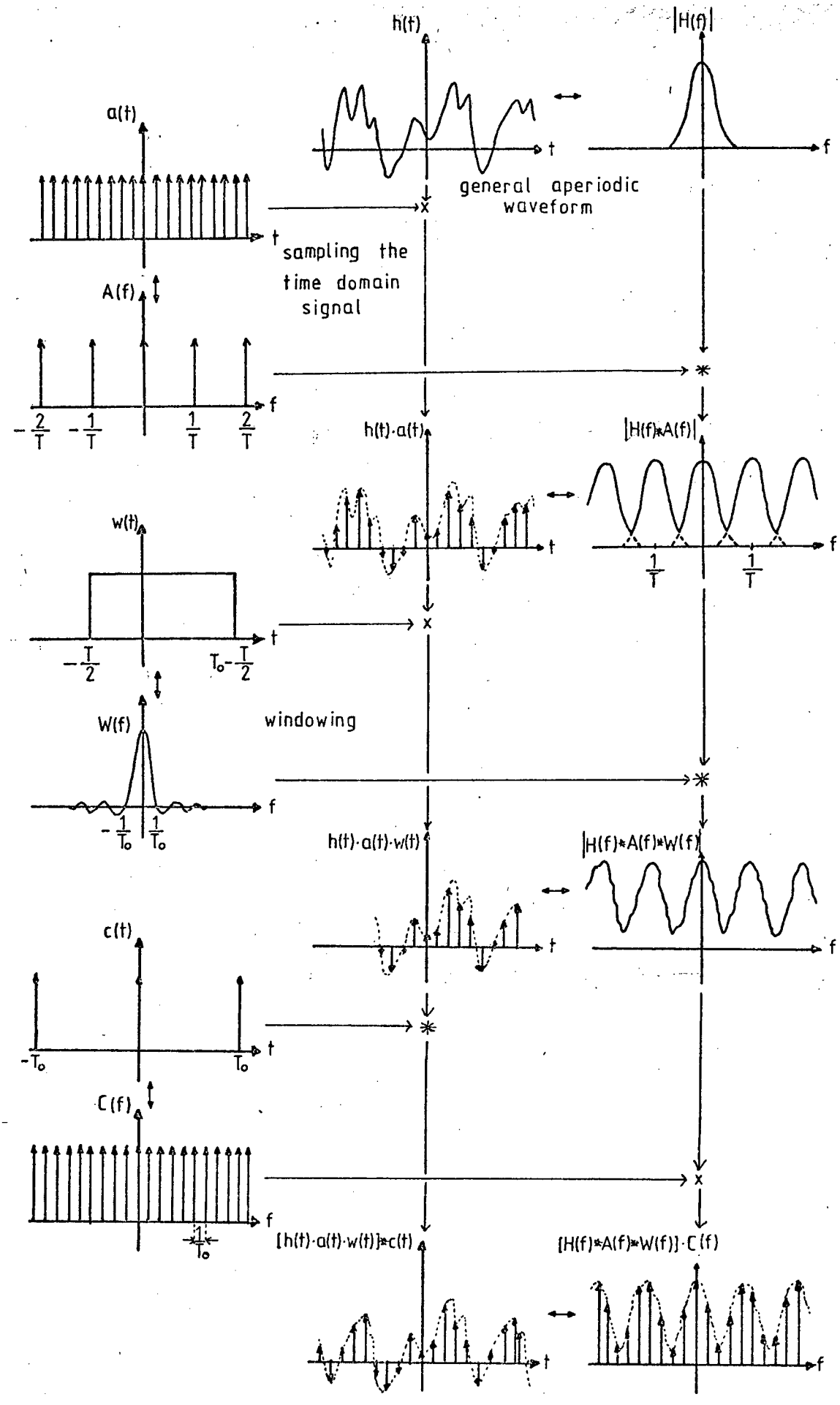


Figure 2.4 Development of the discrete Fourier transform.

The Cosine, Blackman, Tukey, Papoulis, Parzen, Kaiser and Dolph-Chebyshev are other window functions, each with their own various advantages and disadvantages.

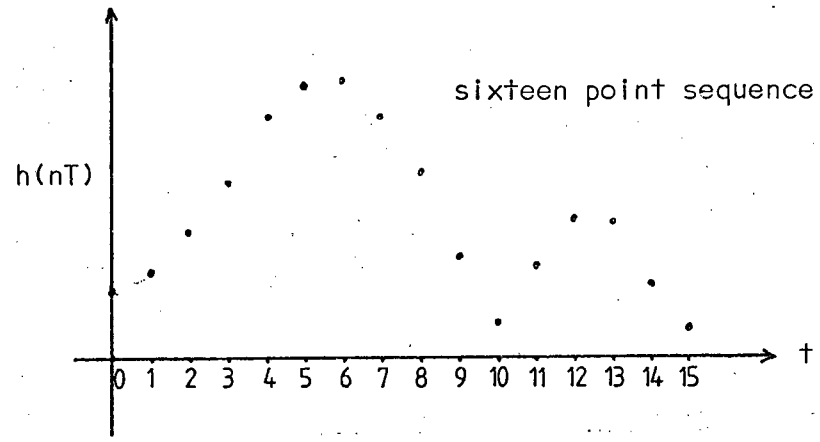
2.3 THE DISCRETE FOURIER TRANSFORM

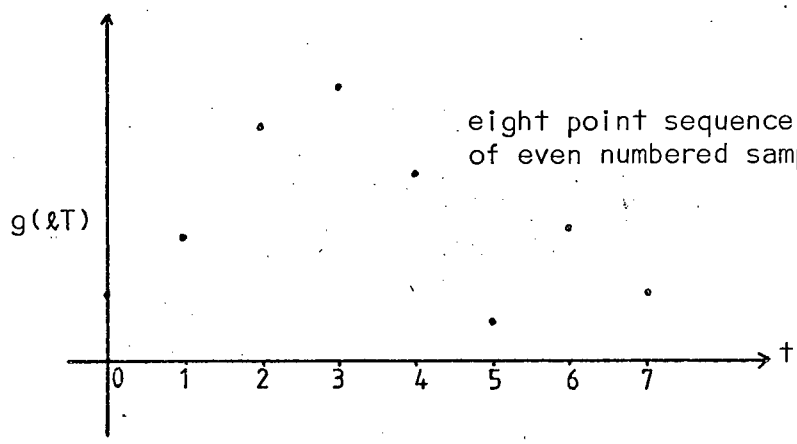
Let h(t) be represented by the sequence of N samples h(nT), $0 \leq n \leq N-1$, where T is the sampling interval in the time domain. Similarly let H(w) be represented by the N samples $H(k\Omega)$, $0 \leq k \leq N-1$ where Ω is the increment between samples in the frequency domain. The discrete Fourier transform may be written as an adaptation of equation 1.1:

$$H(k\Omega) = \sum_{n=0}^{N-1} h(nT).\, e^{-j\Omega Tnk} \qquad k = 0,1,2 \ldots N-1 \qquad \ldots 2.1$$

where $\Omega = \frac{2\pi}{NT}$

The graphical development of the discrete Fourier transform (DFT) along with the effects that sampling, aliasing and windowing produce is shown in Figure 2.4. Sampling the original waveform, the first operation shown in Figure 2.4, may give rise to an aliased frequency function. The next operation to be performed is time domain truncation and this introduces ripple into the frequency domain function. The final operation is frequency domain sampling which causes the time domain





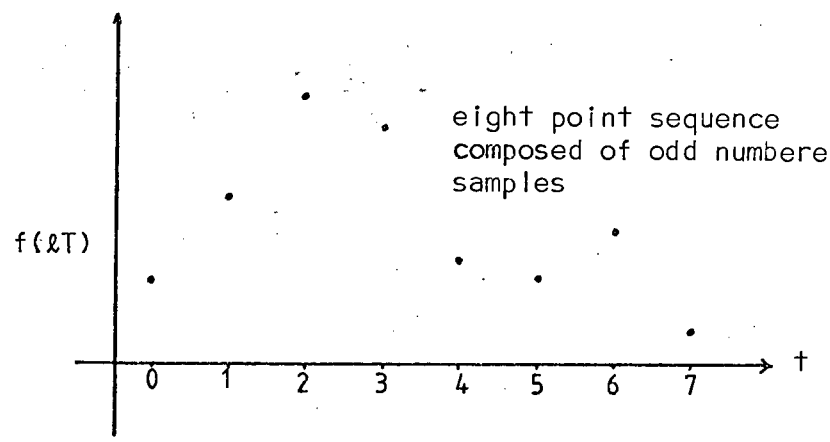


Figure 2.5 Splitting the sixteen point sequence into two eight point sequences.

function to be periodic with a period defined by the N points of the original function after sampling and truncation.

2.4 THE FAST FOURIER TRANSFORM

The fast Fourier transform (FFT) is an efficient algorithm for calculating the discrete Fourier transform. To obtain an idea of the increase in efficiency, if N = 1024 (where N is the number of samples representing the data to be transformed), then the computational reduction over the DFT is more than 200 to 1. Almost every textbook relating to signal processing contains some explanation of the development of the FFT so discussion of the subject here will be limited.

The DFT was defined in expression 2.1 to be:

$$H(k\Omega) = \sum_{n=0}^{N-1} h(nT)\, e^{-j\Omega Tnk}$$

where $\Omega = \frac{2\pi}{NT}$ and $k = 0,1 \ldots N-1$

Let $W = e^{-j\Omega T} = e^{-j\,2\pi/N}$ then we may write:

$$H(k\Omega) = \sum_{n=0}^{N-1} h(nT)\, W^{nk}$$

Assuming that we wished to transform the sixteen point sequence shown in Figure 2.5, we may split the sixteen

point sequence into two eight point sequences, where $g(\ell t) = h(2\ell T)$ and $f(\ell t) = h[(2\ell + 1)T]$ where $\ell = 0, 1, 2 \ldots N/2 - 1$.

The DFTs of these two sequences are also N/2 point sequences and may be written thus:

$$G(k\Omega) = \sum_{\ell=0}^{N/2-1} g(\ell T)(W^2)^{\ell k}$$

and

$$F(k\Omega) = \sum_{\ell=0}^{N/2-1} f(\ell T)(W^2)^{\ell k}$$

The DFT of the entire sequence may be written as:

$$H(k\Omega) = \sum_{\ell=0}^{N/2-1} [g(\ell T)W^{2\ell k} + f(\ell T)W^{(2\ell+1)k}]$$

and, rewriting this expression, we get:

$$H(k\Omega) = \sum_{\ell=0}^{N/2-1} g(\ell T)(W^2)^{\ell k} + W^k \sum_{\ell=0}^{N/2-1} f(\ell T)(W^2)^{\ell k}$$

$$= G(k\Omega) + W^k F(k\Omega)$$

Using this method of splitting the original sequence into two shorter sequences reduces the number of calculations required to compute $H(k\Omega)$. The number of operations required to compute $G(k\Omega)$ and $F(k\Omega)$ directly is $(N/2)^2$ for each and combining them to give $H(k\Omega)$ requires N operations giving a total of $N + N^2/2$ operations. Computing $H(k\Omega)$ directly would require N^2 operations so even at this stage a saving is evident. The same

process which was applied to $H(k\Omega)$ may now be applied to $G(k\Omega)$ and $F(k\Omega)$ in turn, giving four four-point sequences, so reducing still further the number of calculations required. Repeating this process until one point DFTs are required reveals the FFT process since the DFT of a single point is the point itself and thus the whole Fourier transform calculation has been reduced to one of a sequence of complex multiplications and additions.

The approach outlined above is known as decimation in time. Several other algorithms which exploit the properties of the data to be transformed have been developed. Brigham in Reference 2.2, develops the FFT as a sequence of matrix factorisations which provides insight into the analysis of the signal flow graph representation of the FFT. When dealing with data records where the number of points $N \neq 2^m$ where m is an integer then N is factorised and the transform is reduced to elementary transforms of the dimension of the lowest factor. This process is described by Singleton in Reference 2.3.

2.5 DISCRETE CORRELATION AND CONVOLUTION

Discrete correlation and convolution do not necessarily require the use of the FFT but in the interests of efficiency the FFT is usually employed.

The correlation of two continuous functions was described previously by equation 1.3

$$\phi_{xy}(t) = \int_{-\infty}^{\infty} x(\theta).y(t+\theta)d\theta$$

When considering the sampled or discrete signals x(nT) and y(nT) ($0 \leqslant n \leqslant N-1$), we may rewrite equation 1.3 such that:

$$\phi_{xy}(kT) = \sum_{n=0}^{N-1} x(nT).y[(k+n)T] \qquad \ldots 2.2$$

Convolution is of more interest since the ultimate aim is to deconvolve two sampled waveforms. The convolution of two sampled waveforms, with reference to equation 1.4 may be written as:

$$y(kT) = \sum_{n=0}^{N-1} g(nT).x[(k-n)T] \qquad \ldots 2.3$$

As for the correlation of two discrete waveforms, it can be shown that the computation of equation 2.3 may be simplified by using the DFT. In Reference 2.4 Hunt proves the discrete convolution theorem using matrix theory. The approach he adopts is to diagonalise the convolution matrix operator and in doing so the eigen values of this operator are shown to be identical to the discrete Fourier transform.

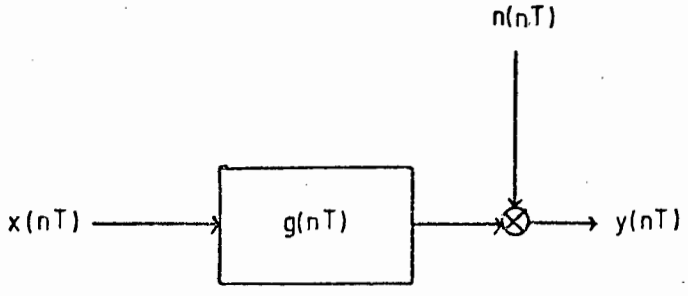


Figure 2.6 The system.

2.6 NUMERICAL DECONVOLUTION

Consider the system with impulse response g(nT), output y(nT) and input x(nT) as shown in Figure 2.6. Neglecting the noise term we have:

$$y(t) = \int_{-\infty}^{\infty} g(\theta).x(t-\theta)d\theta \qquad \ldots 2.4$$

The determination of g(θ) is thus a system identification problem. One approach to the identification of g(nT) is the division of the Fourier transform of the output by the Fourier transform of the input, that is:

$$g(nT) = \frac{\text{FFT } [y(nT)]}{\text{FFT } [x(nT)]} \qquad \ldots 2.5$$

An alternative to the above is a formulation of equation 2.5 using a matrix representation of the variables is given in Reference 2.5 and outlined below.

Let x(t), a continuous function of time, be considered constant between sampling periods. Each sample of x(t) takes the value at the beginning of the sampling period, that is:

$$x(t) \simeq x(nT) \qquad \text{for } nT < t < (n+1)T$$

Similarly assuming g(t) is constant over the sample interval but with the value at the midpoint of the interval so that:

$$g(t) \simeq g\left(\frac{2n+1}{2} \cdot T\right) \qquad \text{for } nT \leq t < (n+1).T$$

Rewriting equation 2.4 assuming $x(t) = 0$ for $t < 0$:

$$y(t) = \int_{0}^{\infty} g(\theta).x(t-\theta)d\theta$$

and if $t = nT$, then:

$$y(nT) = T. \sum_{i=0}^{n-1} g\left(\frac{2n-1}{2} \cdot T - iT\right) . x(iT)$$

so for $n = 1$, $y(T) = T[g(T/2).x(0)]$

$n = 2$, $y(2T) = T[g(3T/2) \cdot x(0) + g(T/2) \cdot x(T)]$

$$\begin{array}{cc} \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \end{array}$$

$n = N$, $y(NT) = T[g\left(\frac{2n-1}{2} \cdot T\right) \cdot x(0) + \ldots$

$$\ldots + g(T/2).x(N-1)]$$

and setting:

$$\underset{\sim}{y} = \begin{bmatrix} y(T) \\ y(2T) \\ y(3T) \\ \\ y(NT) \end{bmatrix} ; \quad \underset{\sim}{g} = \begin{bmatrix} g(T/2) \\ g(3T/2) \\ g(5T/2) \\ \\ g(2N-1/2 \cdot T) \end{bmatrix}$$

$$X = \begin{bmatrix} x(0) & 0 & 0 & \cdot & \cdot & \\ x(T) & x(0) & 0 & & & \\ x(2T) & x(T) & x(0) & & & \\ \cdot & x(2T) & x(T) & x(0) & & \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ x[(N-1).T] & x[(N-2).T] & \cdot & \cdot & \cdot & x(0) \end{bmatrix}$$

and we may write:

$$\underset{\sim}{y} = T.X.\underset{\sim}{g}$$

We now need to solve for $\underset{\sim}{g}$ in the above equation.

If $x(0) \neq 0$ (a finite shift in time may be required to achieve this), then $\det X = [x(0)]^N \neq 0$ and X is non-singular. So:

$$\underset{\sim}{g} = \frac{1}{T} . X^{-1} \underset{\sim}{y} \qquad \ldots 2.6$$

Thus $\underset{\sim}{g}$ may be computed using matrix methods or alternatively each element of the above matrix equation may be written in terms of the others:

$$h_n = \frac{1}{x(0)} \left\{ \frac{y(nT)}{T} - \sum_{i=1}^{n-1} g_{n-i} . x(iT) \right\}$$

where $h_n = h\left(\frac{2n-1}{2} \cdot T\right)$ and $h_1 = \frac{y(T)}{T.x(0)}$

An iterative procedure may be developed if one considers the system impulse response to be, as a first approximation, equal to the system output. This initial estimate is then corrected by adding the difference between the system output and the convolved current approximation of the impulse response. We may therefore write the recursive relation:

$$\hat{g}_{n+1}(t) = \hat{g}_n(t) + \left\{y(t) - \int_{-\infty}^{\infty} x(t-\theta)\hat{g}_n(\theta)d\theta\right\} \quad \ldots 2.7$$

where $\hat{g}_{n+1}(t)$ is the current estimate of $g(t)$.

Transforming this into the frequency domain we have:

$$\hat{G}_{n+1}(w) = \hat{G}_n(w) + Y(w) - X(w) \cdot \hat{G}_n(w)$$

with $\hat{G}_0(w) = Y(w)$

This technique forms the basis of the work described by Balslev et al. in their paper (Reference 2.6) on the deconvolution of experimentally broadened spectra in the field of spectroscopy. Kennet et al. (in Reference 2.7) also use a similar iterative procedure to achieve improved spectral resolution by applying Bayesian deconvolution.

In the field of seismic exploration several deconvolution methods are being examined with varying degrees of success. Among these are predictive deconvolution, homomorphic deconvolution, Kalman filtering and deterministic deconvolution. An overview of these methods is given in Reference 2.8 where Arya and Holden point out that homomorphic deconvolution has still to be fully developed and they also note that at the time of their writing (1978), there was no reported application of Kalman filtering to seismic deconvolution.

Eisenstein and Cerrato (in Reference 2.9) describe a deconvolution technique which they use to improve electrocardiograms by removing the effects of the measuring system. In their work they also use an iterative technique to improve the estimate of the system impulse response except that a model of the system to be identified is formed and the parameters varied to minimise the error between the real system output and the estimated output.

Several researchers claim success using the Fourier transform techniques described by equation 2.5. Among them are Nabel and Mundry(Reference 2.10) in the field of ultrasonic testing and echo evaluation and Kuchel (in Reference 2.11) in his work on a flow assay device for the study of steady-state enzyme kinetics.

The work of this dissertation will be concentrated on the approach described by equation 2.5.

2.7 APPLYING THE FAST FOURIER TRANSFORM USING A MICROPROCESSOR

The importance of the fast Fourier transform arises due to the fact that most deconvolution methods seem to be based upon the FFT. To obtain some idea of the feasibility of computing the FFT on a microprocessor, the hardware was developed (see Appendix 1) and the task of writing the necessary software was set as an undergraduate thesis co-supervised by me. The hardware comprised a standard Intel SDK85 development kit, 2K of additional random access memory (RAM), 1K of erasable programmable read only memory (EPROM), two eight bit analogue to digital converters and a display control unit which permitted the graphical display on an oscilloscope of data. The software to enable the sampling of an analogue waveform and the subsequent computation of its Fourier transform using the fast Fourier transform was developed (Reference 2.12). The execution time required for a 256 point FFT, using eight bit number representation, was found to be approximately two seconds. Using a sixteen bit number representation, a 256 point transform takes approximately nine seconds. These execution times are excellent considering the very limited arithmetic capabilities of the 8085 microprocessor. It became

apparent that to reduce truncation error to an acceptable level representation of the data using 16 bits was imperative. Computation time would be reduced considerably if a present generation 16 bit microprocessor were employed. The use of a microprocessor to calculate the FFT was successful and a dedicated microprocessor, incorporated into the existing flowmeter, could be used to perform the Fourier transform analysis and possibly the entire deconvolution.

REFERENCES - CHAPTER 2

2.1 Childers, D.G. and Durling, A.E.: Digital Filtering and Signal Processing.
West Publishing Company, 1975, p.254.

2.2 Brigham, E.O.: The Fast Fourier Transform.
Prentice-Hall, 1974, pp.148-153.

2.3 Singleton, R.C.: An Algorithm for Computing the Mixed Radix Fast Fourier Transform.
IEEE Transactions on Audio and Electroacoustics, Vol. AU-17, No. 2, June 1969, pp.93-103.

2.4 Hunt, B.R.: A Matrix Theory Proof of the Discrete Convolution Theorem.
IEEE Transactions on Audio and Electroacoustics, Vol. AU-19, No. 4, December, 1971, pp.285-288.

2.5 Sage, A.P. and Melsa, J.L.: System Identification.
Academic Press, 1971, pp.6-10.

2.6 Balslev, J.E. et al.: Noise Amplication and Resolution Improvement in Deconvolution of Experimental Spectra.
Applied Spectroscopy, Vol. 32, No. 5, 1978, pp.454-457.

2.7 Kennet, T.J. et al.: Bayesian Deconvolution I, II and III. Nuclear Instruments and Methods; 1978, Vol. 151, pp.285-293, pp.293-301 and Vol. 153, pp.125-135 respectively.

2.8 Arya, V.K. and Holden, H.D.: Deconvolution of Seismic Data - An Overview. IEEE Transactions on Geoscience Electronics, Vol. GE-16, No. 2, April 1978, pp.95-98.

2.9 Eisenstein, B.A. and Cerrato, L.R.: Statistical Deconvolution of Electrocardiograms. IEEE Transactions on Biomedical Engineering Vol. BME-25, No. 1, January 1978, pp.96-99.

2.10 Nabel, E. and Mundry, E.: Evaluation of Echoes in Ultrasonic Testing by Deconvolution. Materials Evaluation, January 1978, pp.59-61 and p.77.

2.11 Kuchel, P.W.: Discrimination Between Steady-State Enzyme Kinetic Functions: An Improved Flow Assay Device Using Fourier Deconvolution. Analytical Biochemistry, Vol. 88, 1978, pp.37-61.

2.12 Weeks, D.P.: Microprocessor-Based Fast Fourier Transform.
B.Sc. Thesis, University of Cape Town, 1980.

BIBLIOGRAPHY

Harris, R.W. and Ledwidge, T.J.: Introduction to Noise Analysis.
Applied Physics Series, Pion Ltd., 1974.

Rader, C.M. and Gold, B.: Digital Processing of Signals.
Lincoln Laboratory Publications, 1969.

Otnes, R.K. and Enochson, L.: Digital Time Series Analysis.
John Wiley and Sons, 1972.

Kosina, Z.: A Fast Deconvolution Method for Small Computers.
Radiochemical and Radioanalytical Letters, Vol. 32, 1978, pp.191-196.

CHAPTER 3

SOFTWARE AND EXPERIMENTAL DEVELOPMENT

The actual mechanism of multiphase fluid flow is fairly complex and when considering the generation of flow signals in the presence of a curved velocity profile, the formulation of the problem using a mathematical analysis becomes very difficult. The comparison of the computed velocity distribution and the actual velocity distribution may be made by setting up a flow system from which the flow signals and measurements of the velocity profile are obtained. The flow signals are analysed to find a calculated velocity distribution using the techniques described earlier. This velocity distribution or alternatively the transit time distribution is then compared with the measured velocity or transit time distribution to determine the accuracy and/or feasibility of the proposed techniques.

Investigating methods for finding the velocity distribution in multiphase flow requires the ability to measure and control accurately the velocity profile in the flow of interest. Measurement and control of the velocity profile is by no means a simple matter and this led to the idea of simulating the flow system on a computer. This idea is attractive, bearing in mind that the analysis, in the development stage, will almost certainly be carried

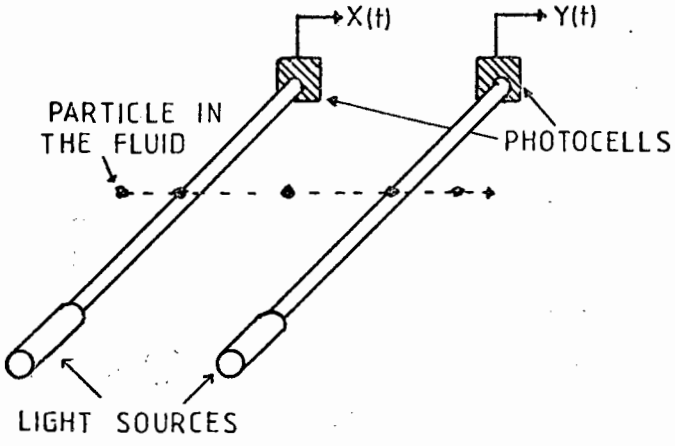


Figure 3.1 Obtaining the flow signals optically.

out on a computer. To eliminate the effects of any assumptions made when modelling the flow system, it would also be desirable to analyse real flow signals.

The development thus consists of three parts:

1. The flow system modelling and flow signal generation;
2. The experimental flow rig; and
3. The analysis of the flow signals.

3.1 FLOW SYSTEM MODELLING AND FLOW SIGNAL SIMULATION

3.1.1 Introduction

The ultrasonic transmitter-receiver pair used to generate the flow signals from variations in the flow each exhibit directivity patterns characteristic of ultrasonic transducers. This gives rise to a complex ultrasonic beam across the pipe. It becomes difficult to determine exactly when any particle or eddy enters the beam. Cross-coupling between the upstream and downstream transducers may also occur. To minimise these complexities optical transducers producing a well-defined beam across the fluid are used. It is reasonable to assume that the fluid contains small particles or bubbles which modulate these light beams to produce the flow signals. For the purposes of modelling this optical

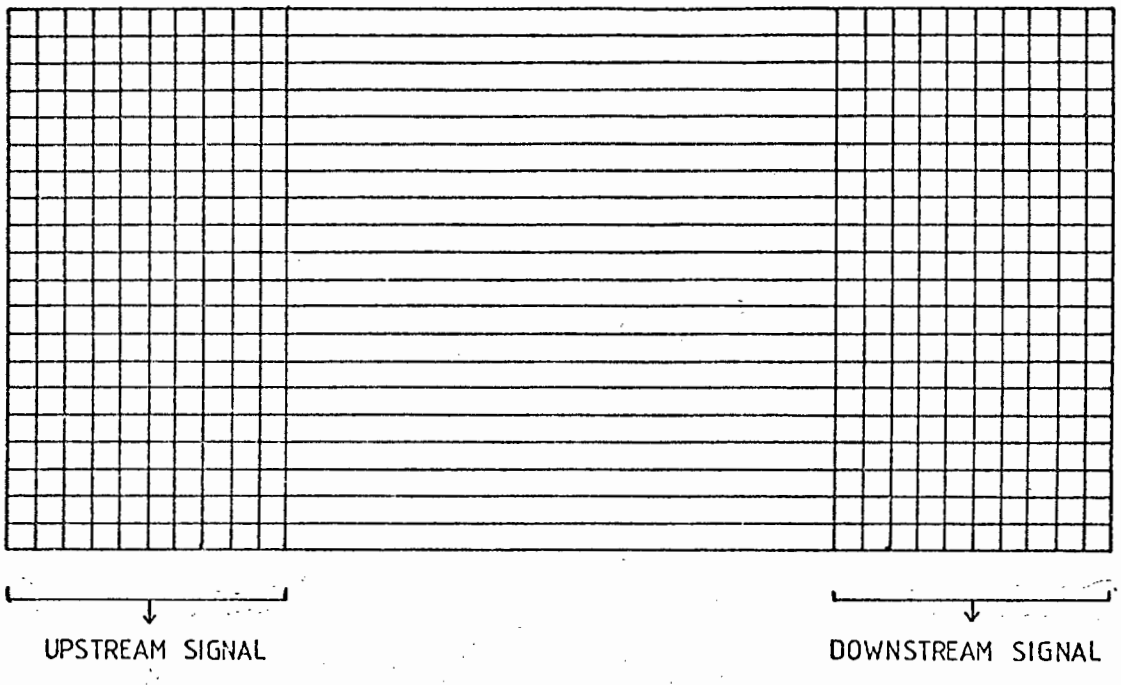


Figure 3.2 The computer representation of the pipe and transducers.

crosscorrelation flowmeter, it is assumed that the beam is wider than the largest particle in the fluid. These particles may then be seen to either scatter or absorb the light with the amount of scattering or absorption depending on the particle size.

3.1.2 Program Specifications

The simulation program is to generate flow signals and calculate the autocorrelation and crosscorrelation functions of the upstream and downstream flow signals. The user should specify the prevailing velocity profile, the transducer spacing and eddy decay time constant.

3.1.3 Program Description

The actual flow signal generation is achieved by using a two-dimensional array as shown in Figure 3.2. The squares of the data array represent the fluid elements of the real pipe. These individual elements are assumed to be travelling at constant velocity. The elements of the array are represented by memory locations of the computer and are filled with normally distributed random numbers to model the random variations in the fluid. The flow signals are then generated by summing the random numbers in the array which are in the field of view of the upstream or downstream tranducers. In Figure 3.2,

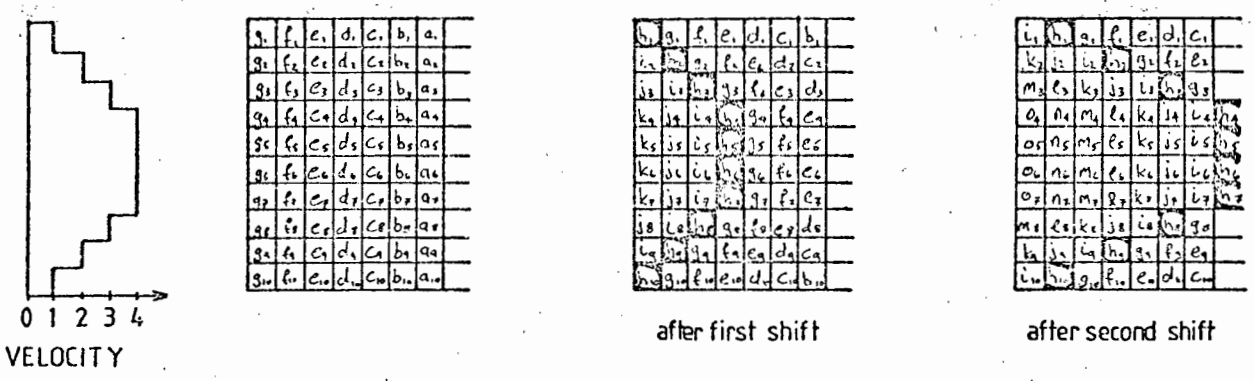
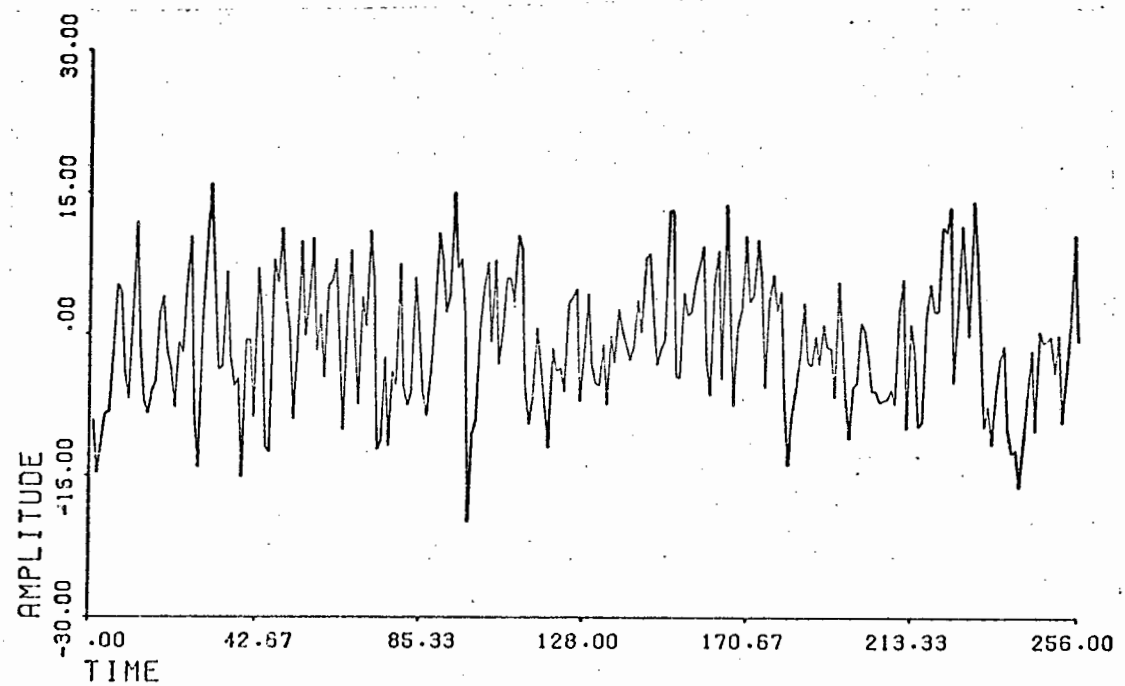


Figure 3.3 An example of the velocity profile effects.



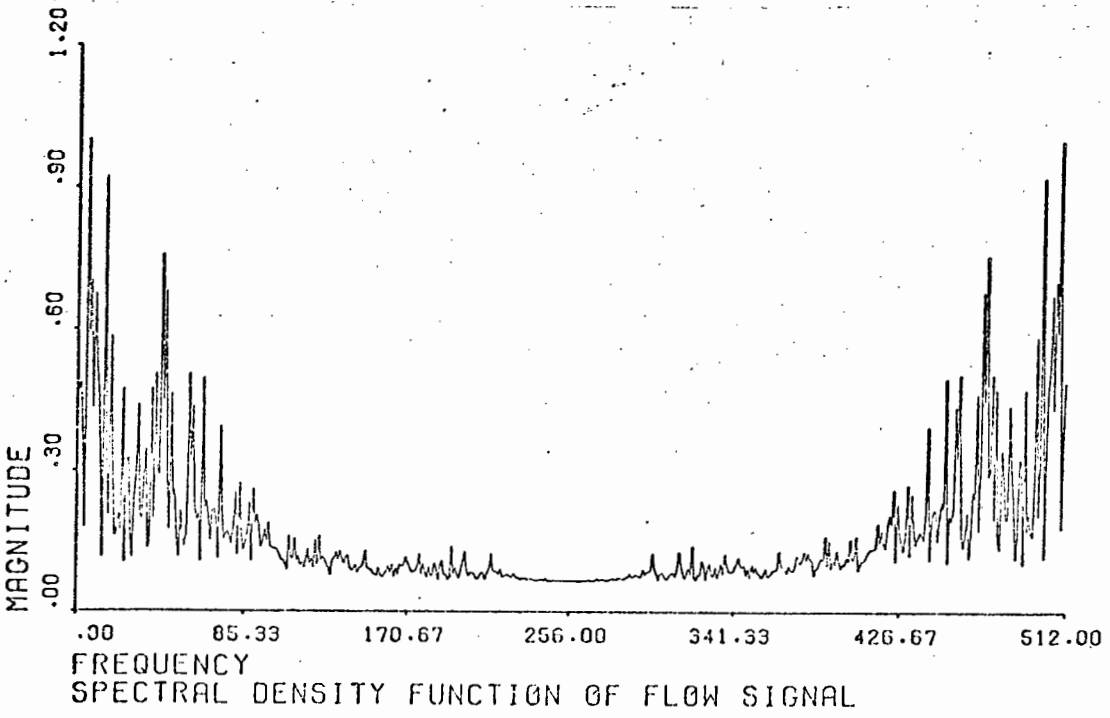


Figure 3.4 A simulated flow signal and its spectral density.

the beamwidth has been chosen to be 10 fluid elements wide. This means that the beamwidth is at least ten times the width of the smallest particle.

The data in the array is shifted along the pipe and the next sample of the flow signals is generated. The number of shifts the fluid elements in the subchannels undergo each time interval is determined by the specified velocity profile. Only the length of pipe between the transducers is of interest so when shifting a row or subchannel along, new numbers are moved in at one end and dropped off at the other. The eddy decay effect is achieved by adding uncorrelated noise to the numbers in the individual fluid elements so that the crosscorrelation coefficient is inversely proportional to transducer spacing. The amount of noise added increases exponentially with time.

3.1.4 Simulation Results

An example of a typical flow signal and its spectral density is given in Figure 3.4. The folding about the centre of the spectrum is a property of the discrete Fourier transform used in the computation of the spectral density (refer to Section 2.3). This spectral density compares well with those given in Reference 3.1. An analysis of this flow signal was carried out to compare the characteristics with those of real flow signals.

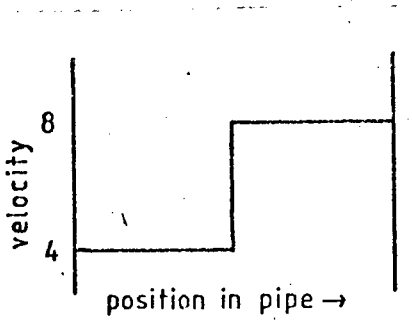


Figure 3.5 The velocity profile.

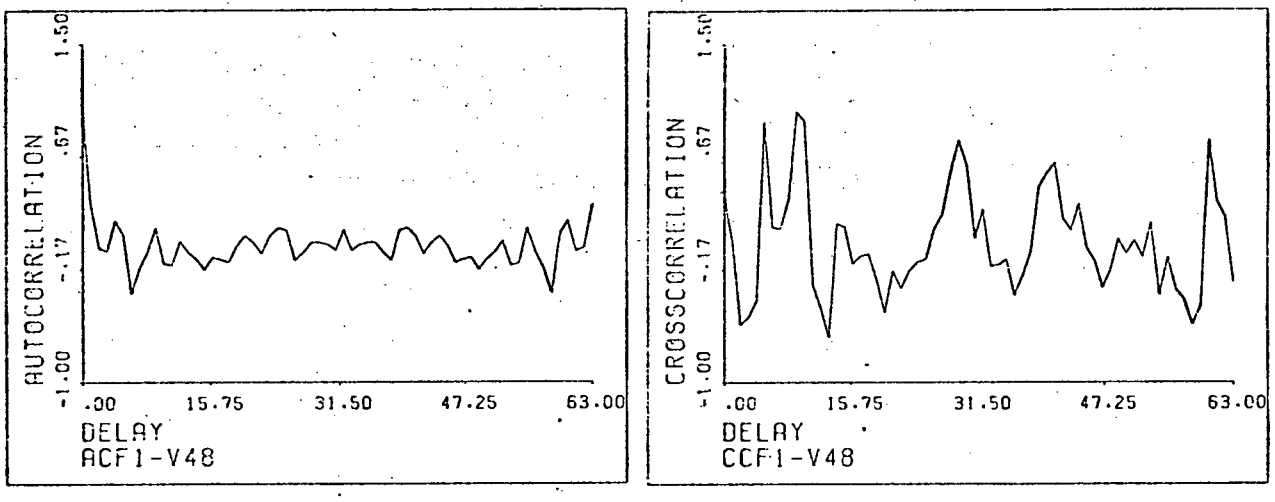


Figure 3.6 The ACF and CCF of the simulated flow signals using only one sample of the flow signals.

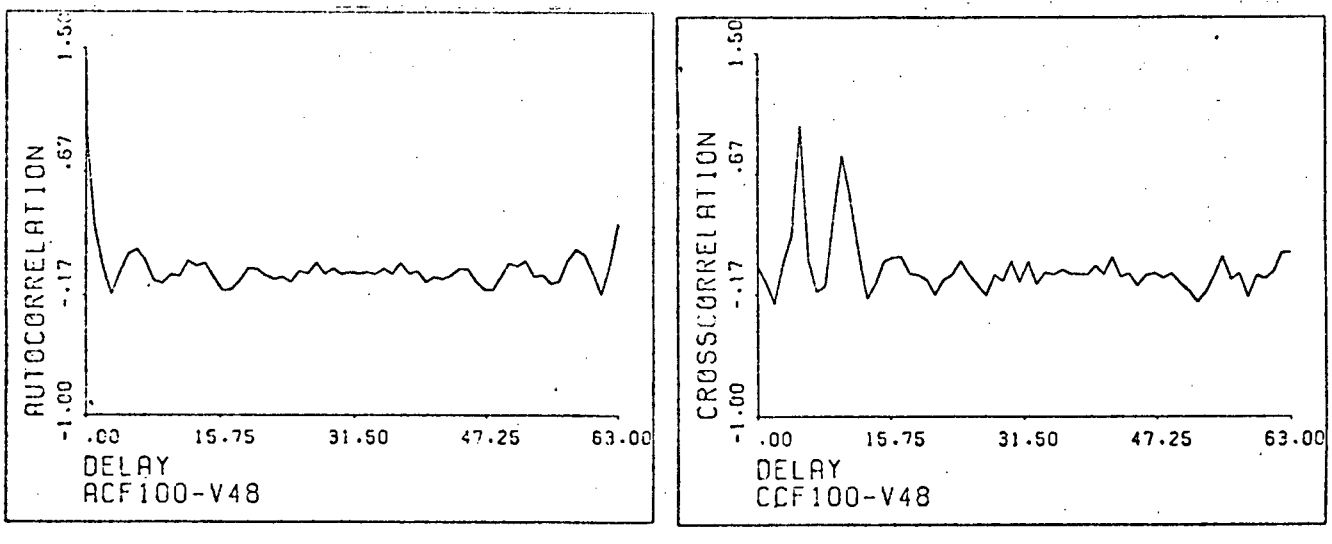


Figure 3.7 The ACF and CCF of the simulated flow signals using 100 samples of the flow signals.

A chi squared goodness of fit test was performed to determine whether the amplitude of the flow signal is normally distributed. The results showed this assumption to be valid at the 50% significance level. One would expect the amplitude to be normally distributed because the signals are simulated by forming a summation of normally distributed random numbers (by the central limit theorem).

The simulation was carried out using an infinite eddy decay time constant and the velocity profile shown in Figure 3.5. The resulting crosscorrelation function (shown in Figure 3.6) is expected to have peaks corresponding only to the two velocities present. The heights of the two peaks should be equal considering the volume of fluid travelling at these two velocities is the same. This crosscorrelation has been computed using only one flow sample and it is felt that a better estimate may be obtained if it were computed from several flow samples and the average taken. Figure 3.7 shows the autocorrelation and crosscorrelation functions computed using 100 flow signal samples and the velocity profile shown in Figure 3.5.

The simulation results using this stepped velocity profile may be compared with the correlation functions obtained from the following simple experiment. The experimental layout is shown schematically in Figure 3.8.

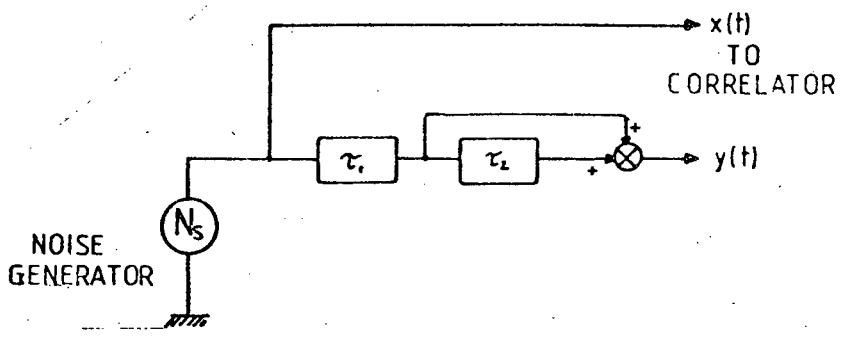


Figure 3.8 The flow system.

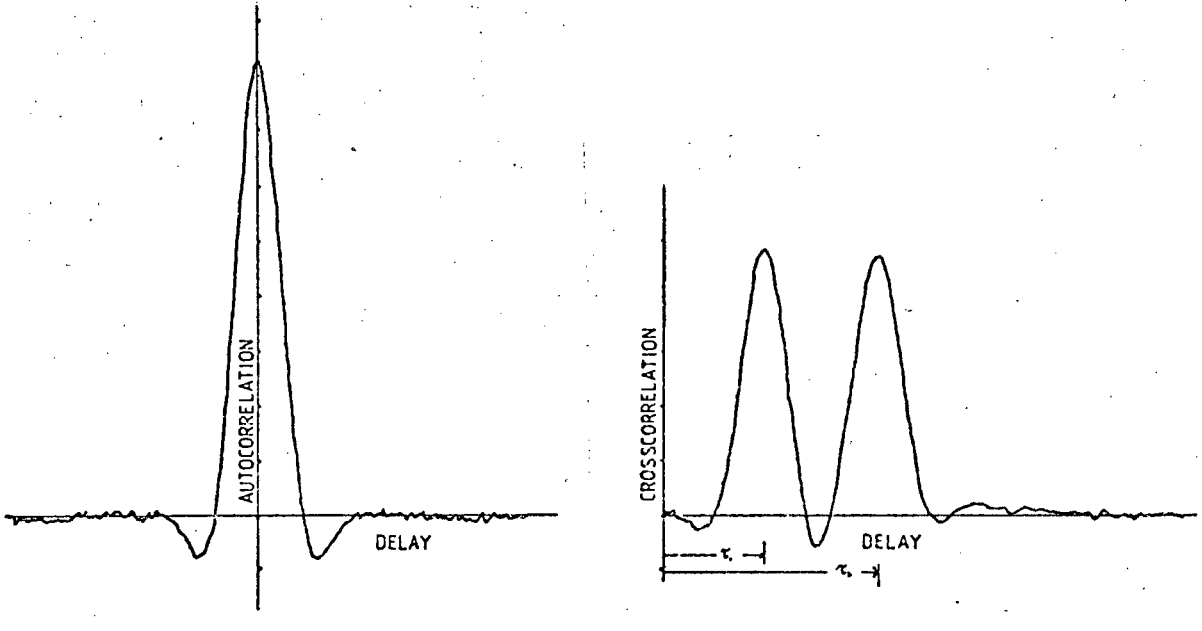


Figure 3.9 The autocorrelation and crosscorrelation functions obtained from the system shown in figure 3.8 above.

The flow signals are generated using a Gaussian noise generator. The delay between the upstream and downstream flow signals and the simulation of the stepped velocity profile is achieved using a tapped delay line. The resulting autocorrelation and crosscorrelation functions are given in Figure 3.9.

The question arises as to the relation between the averaging time or number of flow signal samples used and the variance of the correlation functions. Examining only the autocorrelation function, we may rewrite equation 1.3 with the limits of integration expressed slightly differently:

$$\phi_{xx}(\tau) = \lim_{T \to \infty} \frac{1}{T} \int_0^T x(\theta + \tau).x(\theta)d\theta$$

The averaging time T must be finite for any practical measurement and it can be shown (in Reference 3.2) that the expression for the variance of the autocorrelation estimate of bandlimited white noise of bandwidth B, zero mean value and over an averaging time T is given conservatively by:

$$\text{Var}\ [\hat{\phi}_{xx}(\tau)] \simeq \frac{1}{2BT}\left\{\phi_{xx}^2(0) + \phi_{xx}^2(\tau)\right\}$$

where $\hat{\phi}_{xx}(\tau)$ is the estimated value of the true autocorrelation function $\phi_{xx}(\tau)$. It is further stipulated that for the above expression to hold the following

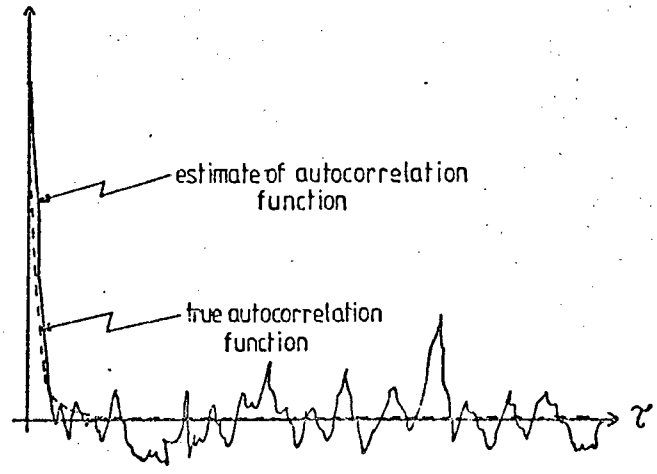
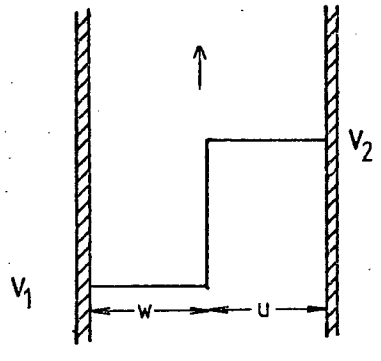


Figure 3.10 The autocorrelation function.



w — width of channel moving at velocity V_1

u — width of channel moving at velocity V_2

Figure 3.11 The velocity profile for simulation.

inequalities should be satisfied.

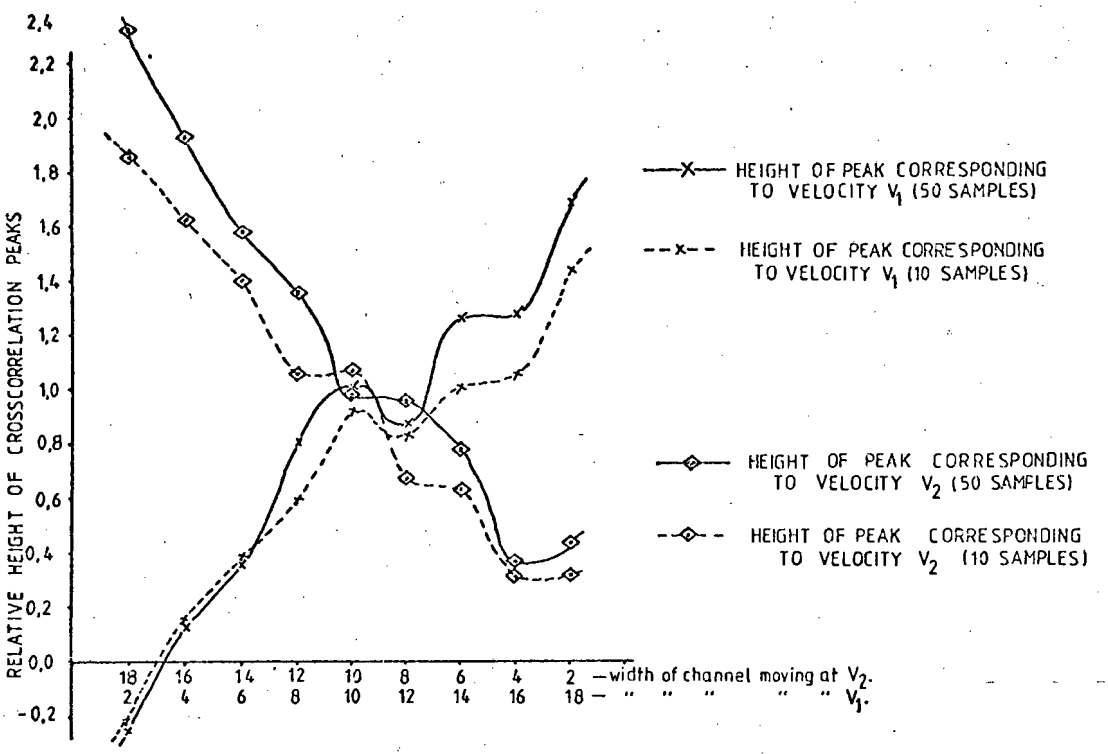
$$BT \geq 5 \quad \text{and} \quad T \geq 10 \cdot |\tau|$$

So for the computer model results to have any statistical significance, it is necessary to average the auto-correlation function over several estimates.

The computing time required to generate two flow signals and to compute the autocorrelation and crosscorrelation functions is approximately three minutes using the program developed. The total time required increases as a linear function of the number of estimates required. Thus assuming the desired variance is 1% of the peak of the autocorrelation function (this implies a standard deviation of 10%), one hundred estimates are needed or approximately three hundred minutes of computer time!

A second and more efficient simulation program (listed in Appendix 2) has been developed. SIMULA V2.0 as it is called, allows the user to specify the number of averages over which the correlation functions are to be calculated. This program also facilitates relatively easy changes of pipe length, pipe width, transducer beamwidth and the number of data representing each flow signal.

Results using the stepped velocity profile shown in Figure 3.11 are in the form of a comparative study.



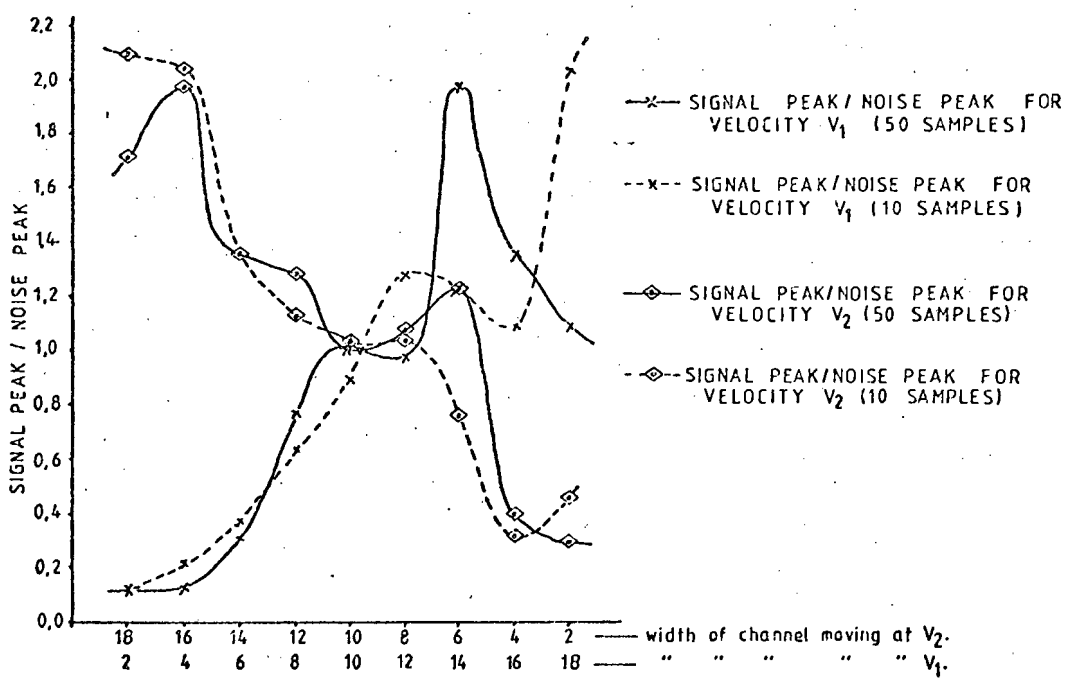


Figure 3.12 The comparative curves.

The relative heights of the peaks corresponding to the velocities V_1 and V_2 are plotted as the width of the subchannel moving at that velocity is varied. The curves are plotted for 10 and 50 averages of the cross-correlation function. A linear relation between the peak height of the crosscorrelation function and the subchannel width is evident if one examines the results given in Figure 3.12. A similar study was carried out plotting the ratio of signal peak to noise peak of the crosscorrelation function as a function of subchannel width. The signal peak is found by recording the height of the crosscorrelation function at the delay corresponding to the velocity of interest, while the noise peak is found by examining the maxima of the crosscorrelation function at delays other than those corresponding to the velocities present.

Further simulation results using different velocity profiles are given in Appendix 2.

3.1.5 Conclusion

The simulated flow signals appear to have similar characteristics as those studied by Leitner (Reference 3.1). The spectral density of the flow signal should really be calculated for several different samples of the flow signal then averaged. This would yield a smoother estimate of the spectral density function. The calculated

autocorrelation and crosscorrelation functions found using the stepped velocity profile (as shown in Figure 3.6) are poor estimates. As the averaging time is increased, the estimate of the autocorrelation and crosscorrelation functions improve as is illustrated by Figure 3.7, where 100 samples were used.

Further work comparing the peak heights of the crosscorrelation function and the ratio of signal peak to noise peak for different subchannel widths reveals the expected linear relations between these parameters. The effect of the averaging on the estimated correlation functions is evident if one compares Figures 3.6 and 3.7.

The results show that accurate simulation of the flow signals and the computation of their autocorrelation and cross-correlation functions requires lengthy, and in this case impractical, amounts of computer time. The simulation exercise has proved very useful in providing a better understanding of the mechanism of flow signal generation. The alternative to the computer simulation is to analyse real flow signals using a correlator to perform online calculation of the autocorrelation and crosscorrelation functions. This process will require less calculation time because the flow signal generation and the correlation calculations are carried out by separate, dedicated devices.

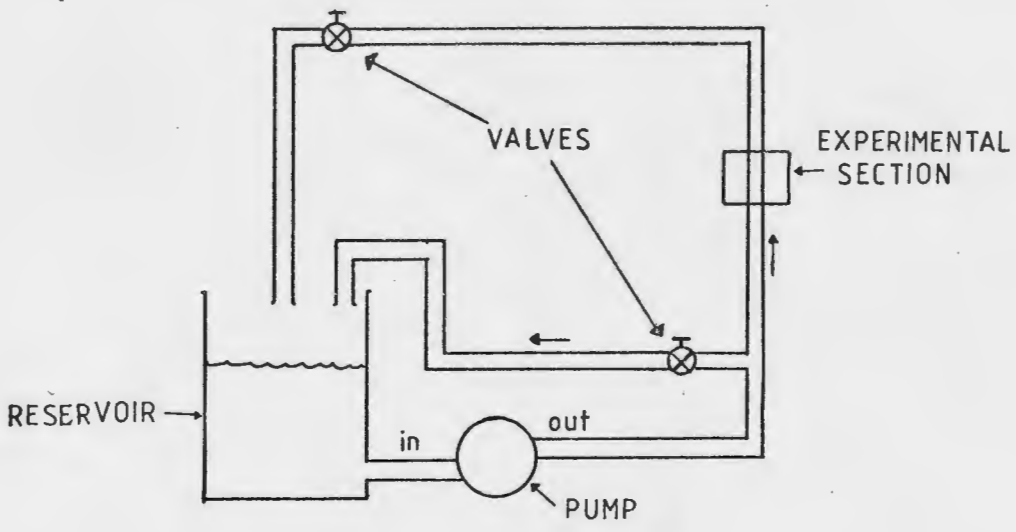


Figure 3.13 A schematic of the flow circuit.

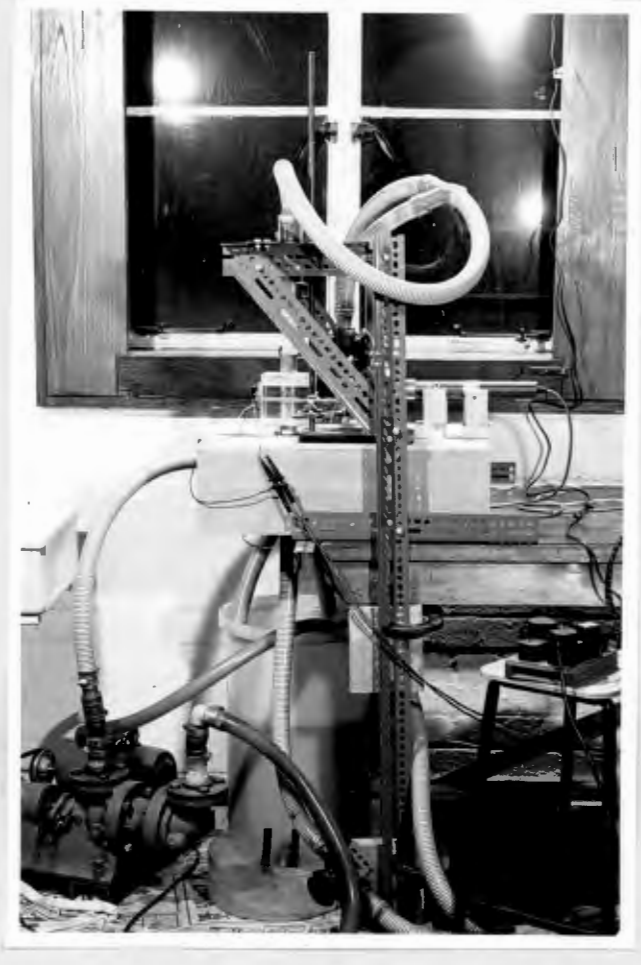


Figure 3.14a The flow circuit and pump.

3.2 THE EXPERIMENTAL FLOW RIG

3.2.1 Introduction

The need for an experimental flow rig arises due to the excessive and impracticable requirements of computer time needed to perform the simulation of the flow signals.

3.2.2 Objectives

The motivation behind this experiment is the desire to measure the velocity distribution in a flow system while computing the autocorrelation and crosscorrelation functions of the upstream and downstream flow signals for the same flow conditions.

3.2.3 Description of the Experiment

The flow rig simply consists of a pump-driven water circuit. This circuit has a section of pipe through which light beams may be passed so that two optically derived flow signals may be obtained and some optical measurement of the velocity distribution made. Some control of the flow velocity is provided by way of two valves and a bypass. Photographs and a schematic are given in Figures 3.13 and 3.14.

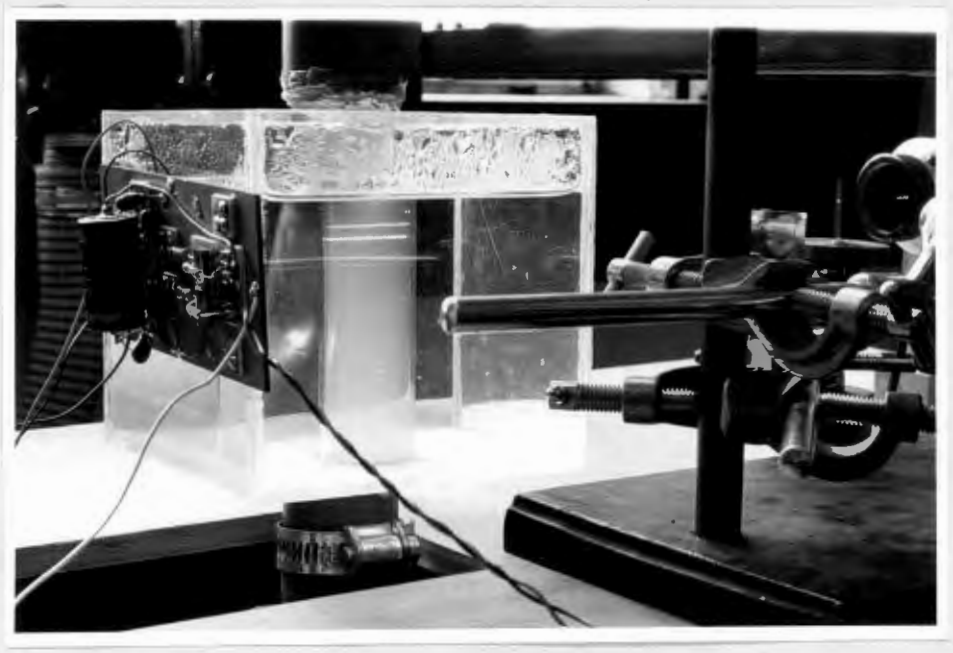


Figure 3.14b A view showing the photocell electronics, the laser, the beamsplitter and the mirror arrangement.

Figure 3.14c A close up showing the parallel laser beams passing through the flow.

The optical system comprises of a helium-neon laser source, a beam splitter and two photocells to produce the flow signals. Particles and bubbles in the water modulate the beams received by the photocells which produce an electrical current proportional to the intensity of the incident light. The square box arrangement around the experimental section is filled with water and this reduces the refraction of the light beams at the air-pipe interfaces. The calculation of the auto-correlation and crosscorrelation functions of the flow signals is then carried out using a Honeywell correlation and probability analyser. Copies of the correlation functions are obtained from the correlator memory via a chart recorder. The correlator has two modes of operation:

(i) A clipped mode, where the flow signals are quantised to one of two levels and

(ii) A full mode where no quantisation is applied.

It has been shown (in Reference 3.3) that the true correlation functions (computed using the correlator in mode (ii)) and those obtained when using the correlator in clipped mode are related by a sine transform. This particular correlator produced better correlation functions when used in the clipped mode.

Finding the velocity distribution requires the measurement of the velocity profile in the flow. The technique used

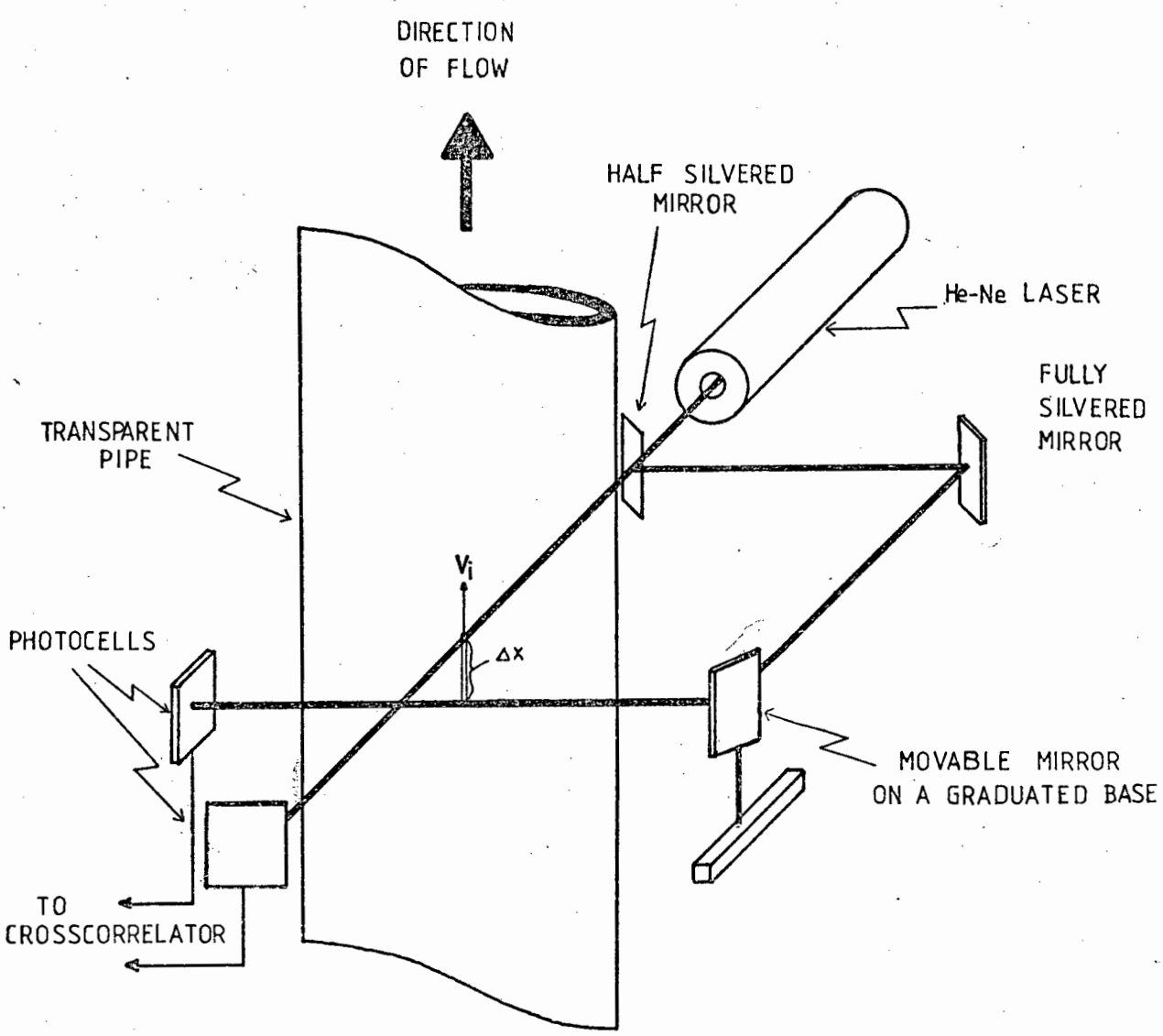


Figure 3.15 Velocity profile measurement using crossed-beam correlation.

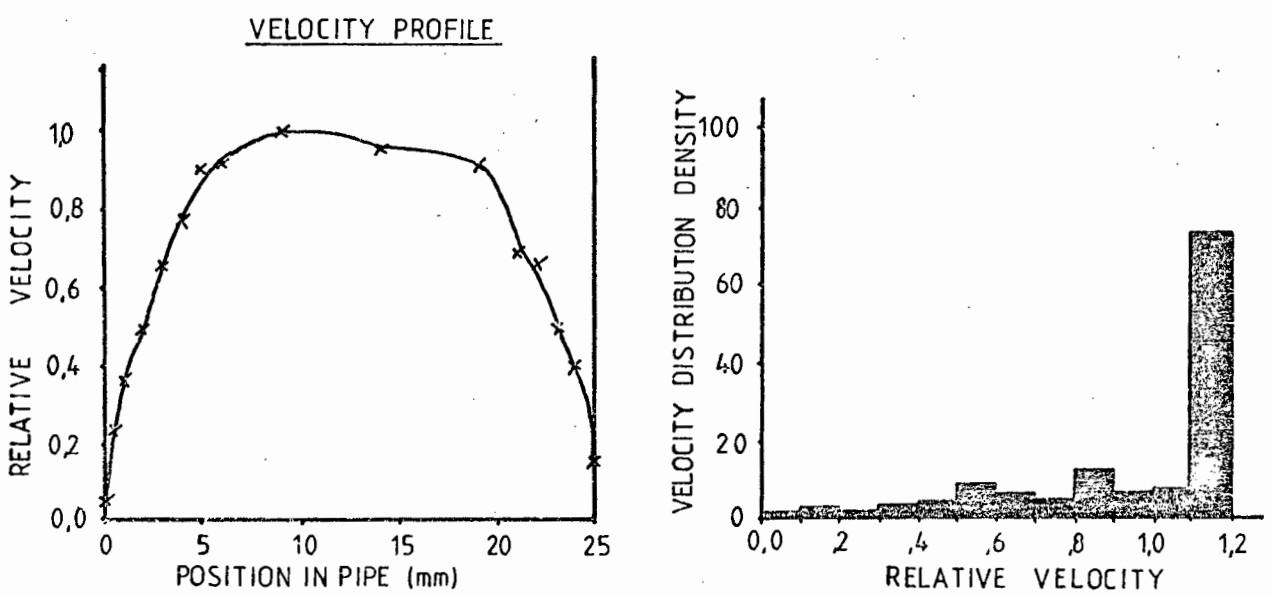


Figure 3.16 The velocity profile and corresponding distribution.

was that of crossed-beam correlation explained in Reference 3.4. This is a completely independent process to that described earlier, although the basic technique remains the same. To achieve crossed-beam correlation, the two laser beams are now set at 90° to one another some small distance apart. This is shown more clearly in Figure 3.15. The correlator is then used to find the time taken for the particles or bubbles travelling in the elemental volume of fluid common to both beams, to move the distance between the laser beams. The velocity profile is obtained by fixing the position of one beam while traversing the other across the pipe diameter.

3.2.4 Results

Sample autocorrelation and crosscorrelation functions are given in Figure 3.17. The velocity profile measured for the same flow conditions is shown in Figure 3.16. The asymmetry of the crosscorrelation function caused by this velocity profile is apparent if one examines Figure 3.16.

3.2.5 Conclusion

The flow rig functions well as may be seen from the results presented. Although this is a more elaborate

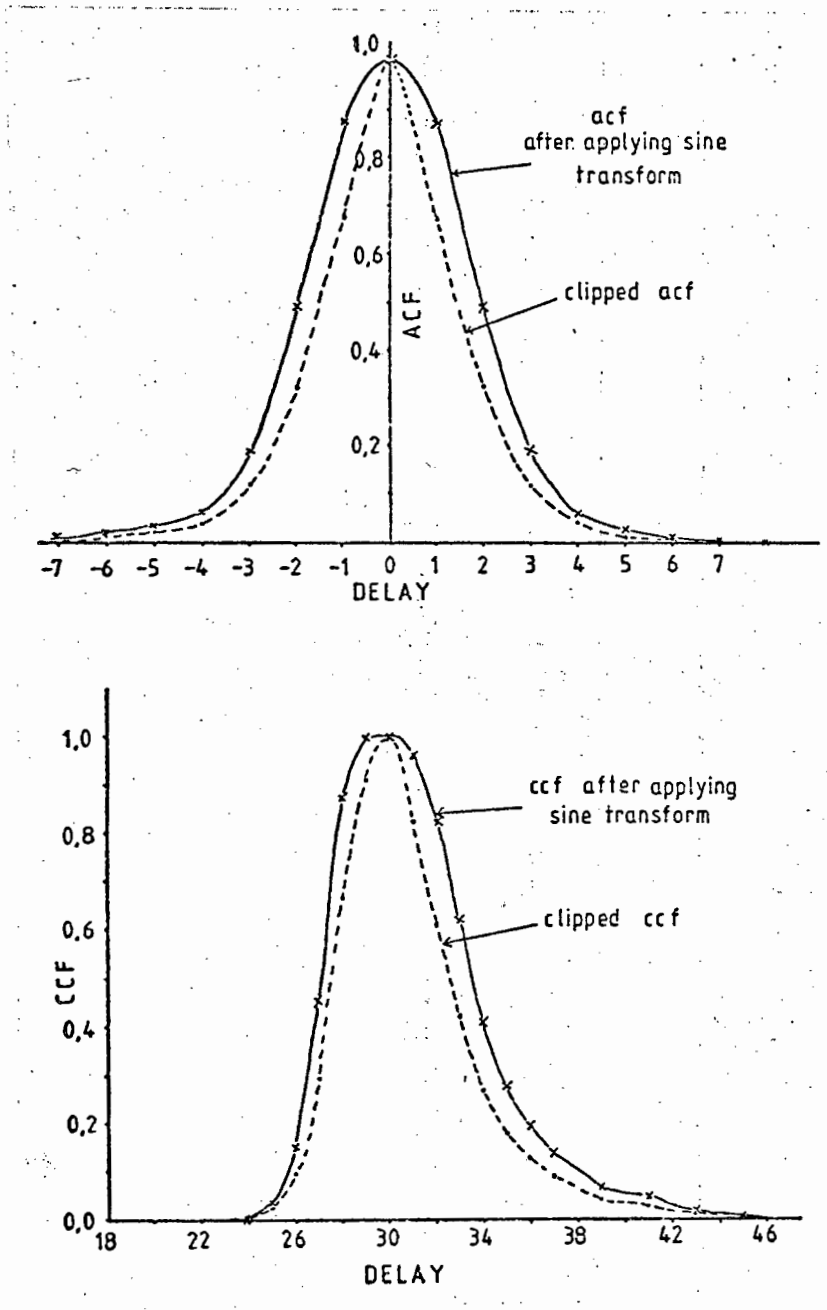


Figure 3.17 The measured autocorrelation and crosscorrelation functions.

method of finding the velocity profile and the auto-correlation and crosscorrelation functions, the results show a definite improvement over the computer simulation results.

3.3 THE ANALYSIS PROGRAM

3.3.1 Introduction

This program has been developed so that deconvolution based on the process described by equation 2.5 may be thoroughly investigated. The package has been developed using a Univac 1106 mainframe computer and the programming language used is Ascii Fortran (level 8R1). The software is designed to be used interactively so that an on line analysis of any waveform may be carried out.

3.3.2 Specifications

The user may operate on any waveform by entering a sampled version of the waveform into one of four work areas. All the operations are then performed on the respective work area. The program enables the user to:

1. Store and recall any waveforms from permanent or temporary memory.
2. Set up labels for the various waveforms which have been temporarily stored.

3. List these labels.
4. Set up labels for each work area.
5. List these work area labels.
6. Compute the Fourier transform of a complex waveform.
7. Compute the inverse Fourier transform of a complex waveform.
8. Apply one of five different data windows to any waveform.
9. Perform polar to rectangular and rectangular to polar conversions on a complex waveform.
10. Fold a waveform about the central point.
11. Convolve two real waveforms.
12. Set parts of any work area equal to a constant value.
13. Normalise any waveform.
14. Apply sine and arcsine transforms to any waveform.
15. Shift a waveform cyclically.
16. Equate various elements of any work area.
17. Examine and change individual data values of each work area.
18. Perform step by step deconvolution of two complex waveforms using the division of the Fourier transforms.
19. List all data values in any work area on the visual display unit (vdu) screen.
20. Print data values on the line printer.
21. Plot the data values of any work area on the vdu screen using a character graph.
22. Plot the data values of any work area using the Calcomp plotter.

23. Add white noise of variable standard deviation and amplitude to any waveform.
24. Perform most arithmetic operations on any work area.
25. Perform arithmetic operations with the work areas as variables.
26. Compute the sum of all the elements in any work area.
27. Correlate two waveforms.
28. List all the commands available.

3.3.3 Description

The basic method of operation is to enter correlation functions obtained either from a computer simulation or experimentally via the terminal keyboard. Operations from those listed above may then be applied to the waveforms to obtain some idea of the usefulness of the proposed analysis techniques. The main program, the various subroutines and some description are given in Appendix 3. Many of the techniques used have been described in Chapter 2.

3.3.4 Results

Almost all the subsections of this program are used to produce many of the results given in Chapter 4. Examples of typical Fourier transforms, inverse Fourier transforms, the data windows, the convolution and deconvolution of

two functions, data listing and the noise function are given in Appendix 3.

3.3.5 Conclusion

This package is a very powerful tool for the application of online waveform analysis and the testing of signal processing techniques. It should not be too great a task to modify or add to the package any particular function seen to be useful.

3.4 SUMMARY

This chapter has dealt with the proposed computer simulation of the flow mechanism and the flow signal generation with a description of the model used. The problems which the computer simulation raises are discussed along with the results. Due to the simulation failure, an experimental flow rig has been developed and sample measurements of the various parameters given. The analysis of the measured parameters has been made possible by the development of a waveform analysis package described briefly here. The detailed working is given in Appendix 3 and the fundamental theory in Chapter 2. Analysis of the parameters measured and some discussion of further investigation into the deconvolution method forms the following chapter.

REFERENCES - CHAPTER 3

3.1 Leitner, J.R.: Slurry Flowmetering using Correlation Techniques.
Ph.D. Thesis, University of Cape Town, 1979, p.183.

3.2 Bendat, J.S. and Piersol, A.G.: Random Data: Analysis and Measurement Procedures.
Wiley-Interscience, 1971, pp.181-184.

3.3 Leitner, J.R.: Op.cit. pp.89-95 and pp.268-278.

3.4 Fisher, M.J. and Krause, F.R.: Crossed-Beam Correlation Technique.
Journal of Fluid Mechanics, Vol. 28, 1969, pp.705-717.

BIBLIOGRAPHY

Hartley, M.G. (Editor): Digital Simulation Methods.
Peter Peregrinus, 1975.

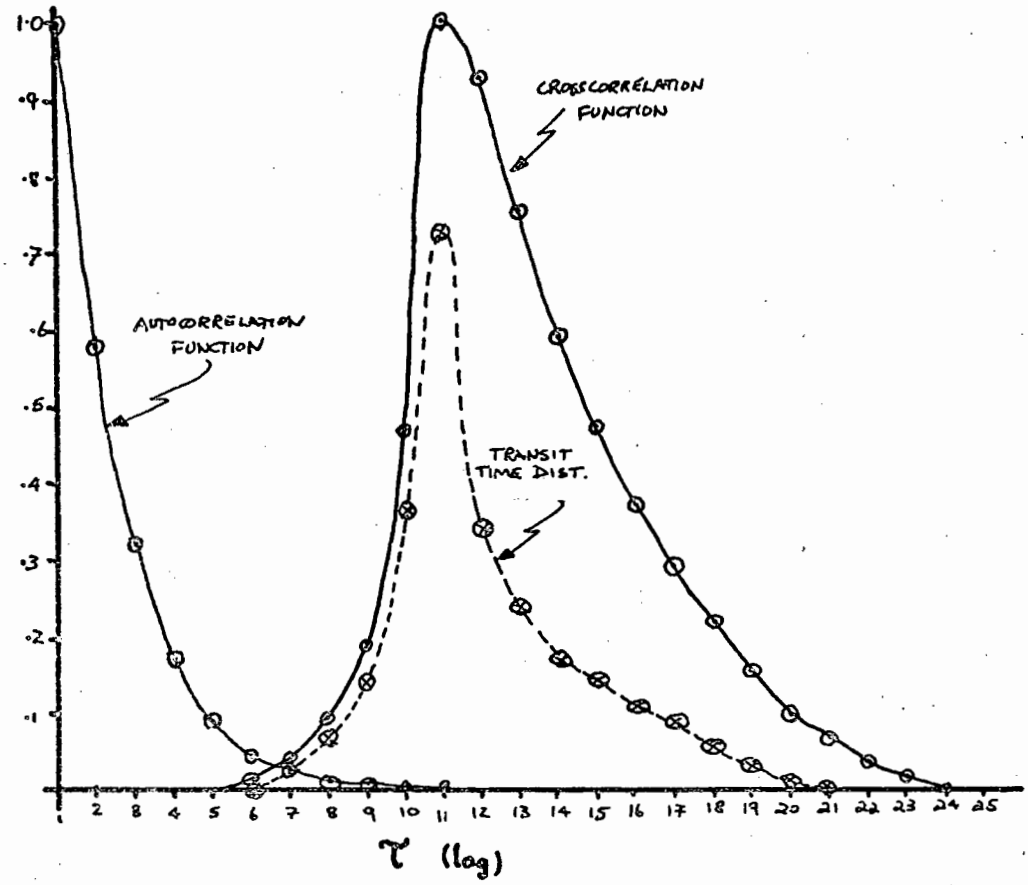


Figure 4.1 The deconvolution of an arbitrary ACF and CCF.

DECONVOLUTION

The results of the deconvolution of simulated autocorrelation and crosscorrelation functions and those measured from the experimental flow rig are examined in this Chapter. Some discussion as to the success or failure of the method in each case is presented at the end of the Chapter.

4.1 DECONVOLUTION OF ARBITRARY AUTOCORRELATION AND CROSSCORRELATION FUNCTIONS

Prior to the development of the simulation and the analysis programs (SIMULA and WAVEPACK), the deconvolution of two arbitrary autocorrelation and crosscorrelation functions was carried out. These two functions as well as the resulting transit time distribution are shown in Figure 4.1.

Given a system with its input signal x(t), output signal y(t) and impulse response g(t), the system output is related to the input by the convolution of the input and the system impulse response. It can be shown (refer to Appendix 4) that if the input signal x(t) is replaced by the autocorrelation of the input, namely $\emptyset_{xx}$, and y(t) is replaced by the crosscorrelation of the input and

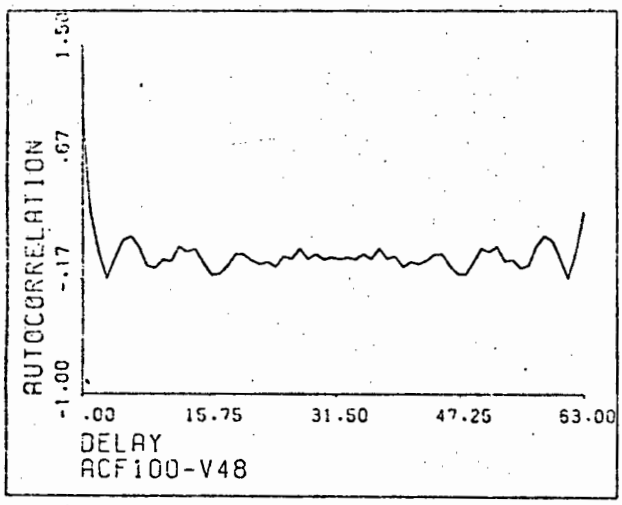


Figure 4.2a The autocorrelation function.

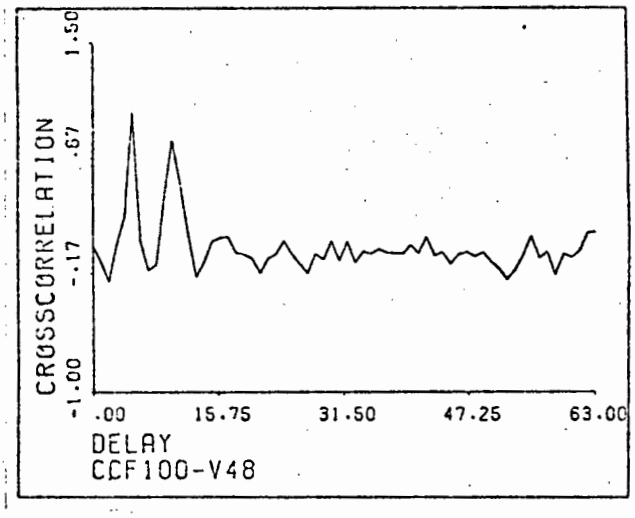


Figure 4.2b The crosscorrelation function.

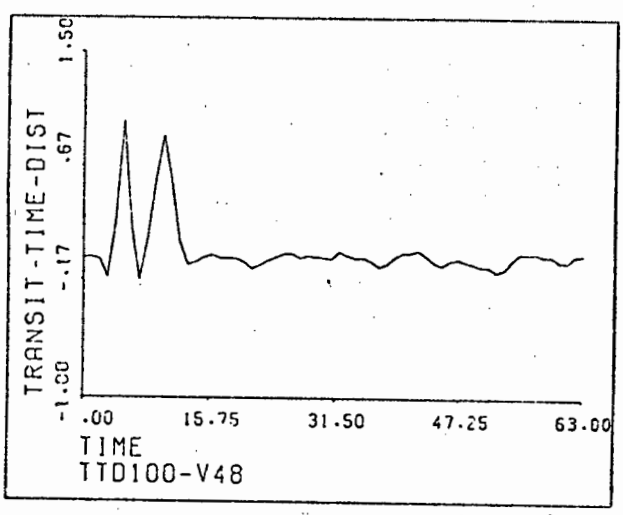


Figure 4.2c The transit time distribution.

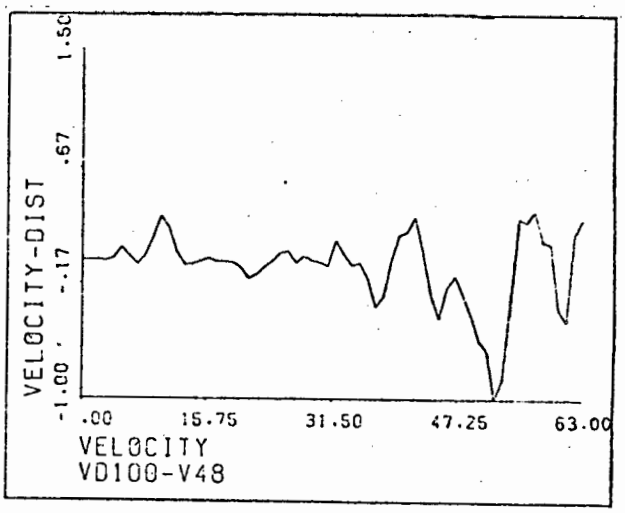


Figure 4.2d The velocity distribution.

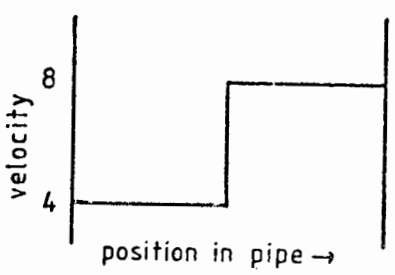


Figure 4.2e The velocity profile.

output signals, namely $\emptyset_{xy}$, then these correlation functions may be used instead of x(t) and y(t) in the convolution relation which still holds.

Considering now the system with its input the autocorrelation function (ACF), output the crosscorrelation function (CCF) and the transit time distribution (TTD) as its impulse response, the effect of the TTD is to smear the CCF. The deconvolution allows one to evaluate this smearing function, the transit time distribution. The results shown in Figure 4.1, although somewhat qualitative, look promising.

4.2 DECONVOLUTION OF AUTOCORRELATION AND CROSSCORRELATION FUNCTIONS OBTAINED FROM THE COMPUTER SIMULATION OF THE FLOW

The autocorrelation and crosscorrelation functions shown in Figure 4.2 were calculated from flow signals simulated using the stepped velocity profile shown in Figure 4.2e. The correlation functions have been averaged over 100 samples. The CCF is expected to exhibit two peaks corresponding to the two velocities present in the velocity profile. The noise due to the finite averaging time is clearly evident if one examines the CCF. The transit time distribution obtained by the deconvolution of the ACF and CCF is shown as well as the velocity distribution obtained by applying

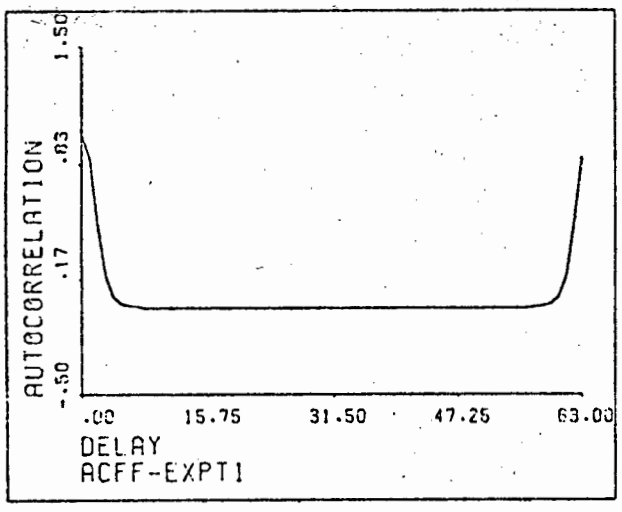


Figure 4.3a The autocorrelation function.

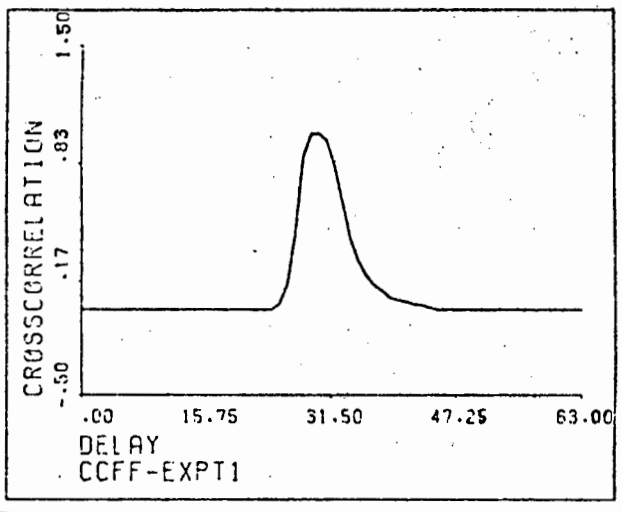


Figure 4.3b The crosscorrelation function.

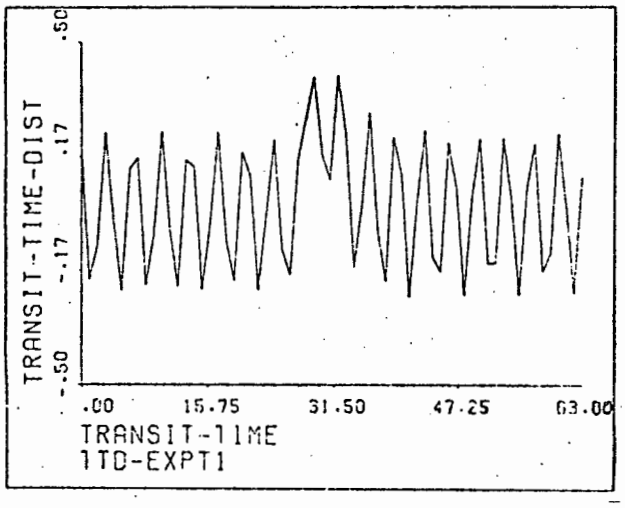


Figure 4.3c The transit time distribution.

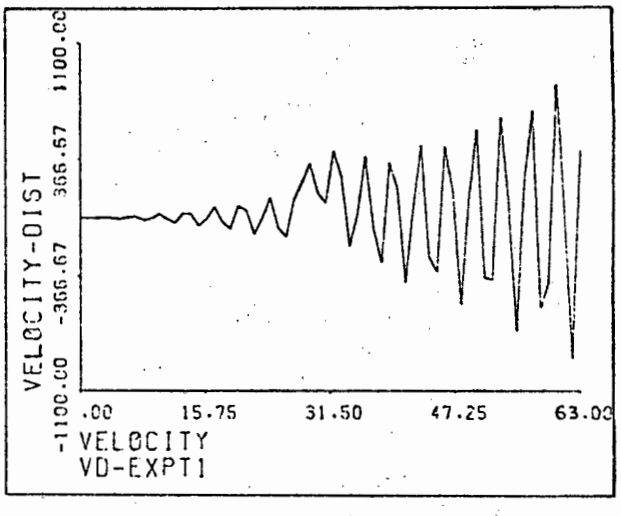


Figure 4.3d The velocity distribution.

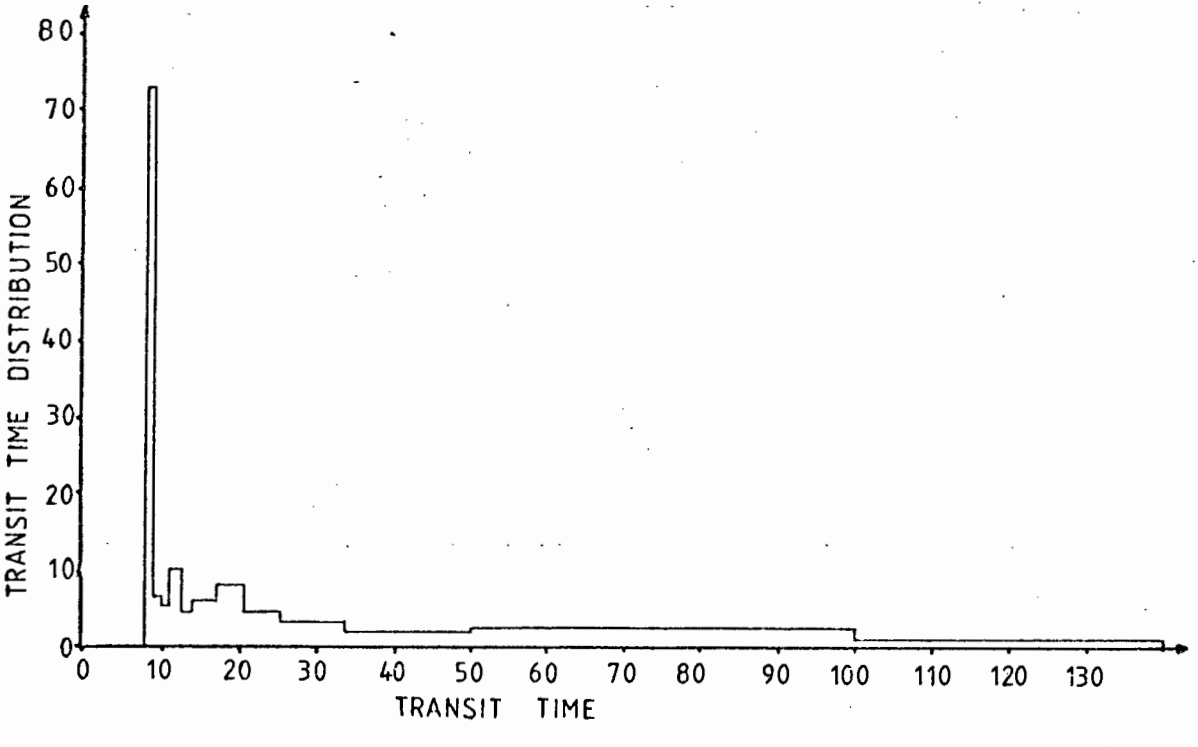


Figure 4.3e The measured transit time distribution (obtained from velocity profile measurements, refer to section 3.2).

equation 1.5. Insufficient averaging time gives rise to the irregular shape of the deconvolved transit time distribution.

Results using flat and stepped velocity profiles and the subsequent deconvolutions are given in Appendix 5.

4.3 DECONVOLUTION OF THE AUTOCORRELATION AND CROSSCORRELATION FUNCTIONS MEASURED FROM THE EXPERIMENTAL FLOW RIG

The ACF and CCF measured from the flow rig are given in Figures 4.3a and 4.3b. The calculated transit time and velocity distributions are given in Figures 4.3c and 4.3d. There is very little similarity between the calculated and expected transit time distributions. Examining the Fourier transforms of the ACF and CCF at two frequencies, the numerical values of the Fourier transform of the ACF are very much smaller (approximately 2000 times) than the corresponding values of the Fourier transform of the CCF. Consequently, when dividing these two transforms, the resulting function has two sharp peaks at these frequencies. When inverse transforming these peaks give rise to the periodic component displayed by the TTD. Various methods of filtering were tried to remove these transmission zeros, but with no success.

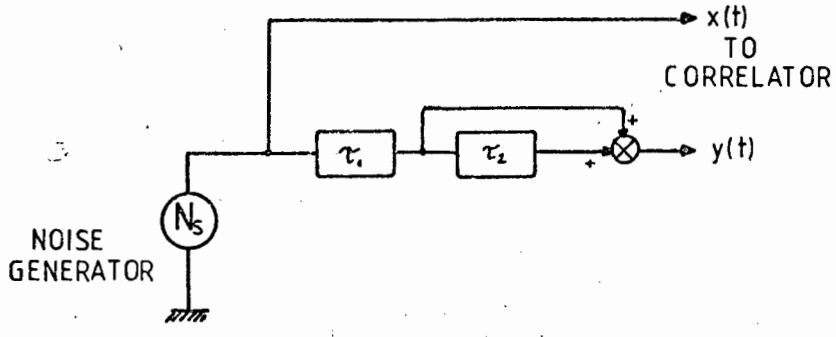


Figure 4.4 Schematic of the simplified flow system showing how the two delays τ_1 and τ_2 are modelled.

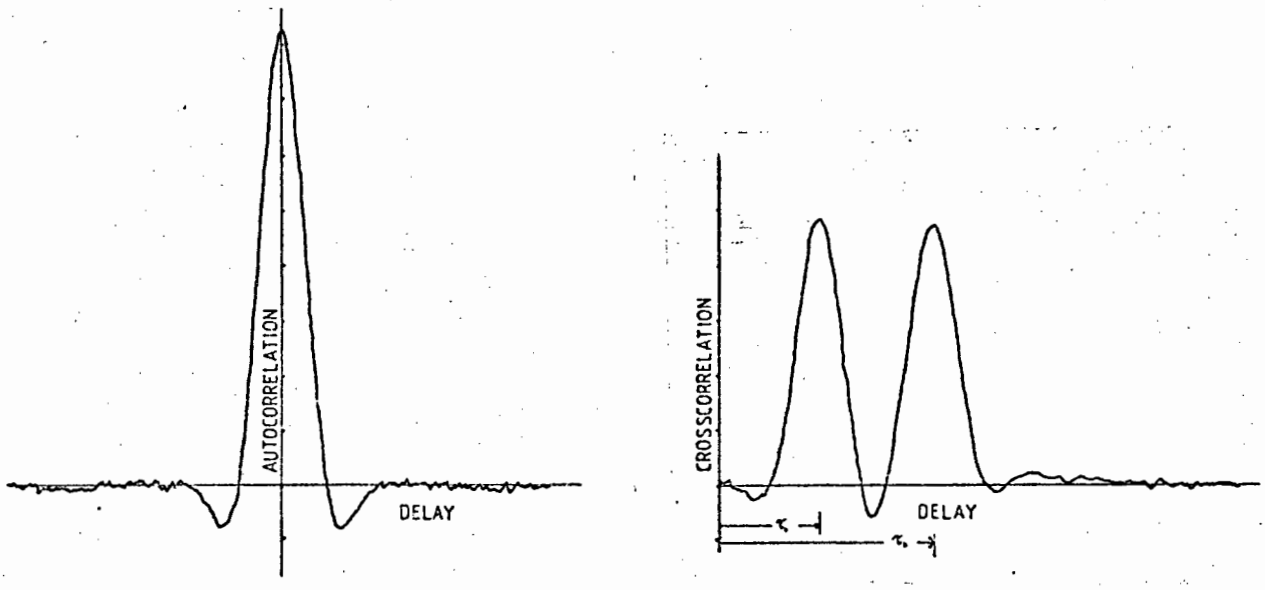


Figure 4.5 The measured autocorrelation and crosscorrelation functions.

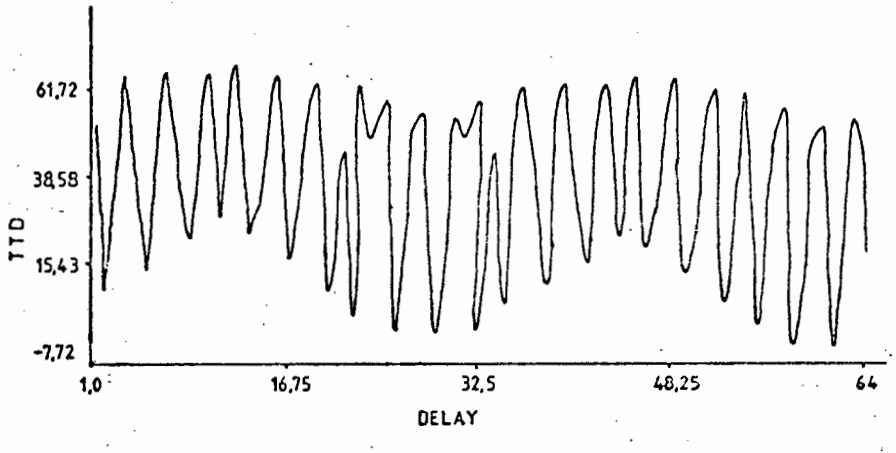


Figure 4.6 The deconvolution result.

4.4 A FURTHER DECONVOLUTION EXPERIMENT

In order to gain further understanding of the deconvolution process, it was decided to experiment with a simplified model of the flow system. This model, shown schematically in Figure 4.4, has a system impulse response or transit time distribution consisting of two delayed unit impulses. These conditions are identical to those produced by pipe or channel flow in the presence of a stepped velocity profile. The aim of this experiment is to deconvolve the autocorrelation and crosscorrelation functions calculated from these "flow signals" in an effort to arrive at the known transit time distribution.

A random noise generator was used as the upstream flow signal and the downstream flow signal was produced using a tapped analogue delay line as the flow process. The downstream flow signal was actually produced by using the upstream flow signal as an input to the delay line and summing the output from two different taps along the delay line.

The ACF and CCF (shown in Figure 4.5) were measured using the correlator in full mode so that the application of the sine transform was not necessary. The deconvolved TTD shown in Figure 4.6 also exhibits the periodicity resulting from noise present in the crosscorrelation function.

4.5 AN ANALYSIS OF THE NOISE EFFECTS ON THE DECONVOLUTION RESULT

Given the linear system with input x(t), output y(t) and impulse response g(t), then we may write, as before:

$$y(t) = \int_{-\infty}^{\infty} g(t-\tau)\, x(\tau)\, d\tau$$

Using the Fourier transform domain to simplify the analysis, we can show that:

$$G(w) = \frac{X(w)}{Y(w)}$$

Now, when dealing with deterministic and noise-free functions the above relations describe the deconvolution process very well. Inevitably the waveforms of interest have some added noise and these expressions must be amended to reflect this. Basically, the expression relating the input and ouput in the above equation must be modified:

$$y(t) = \int_{-\infty}^{\infty} g(t-\tau).x(\tau)\, d\tau + n(t) \qquad \ldots 4.1$$

so that:

$$G(w) = \frac{Y(w)}{X(w)} - \frac{N(w)}{X(w)} \qquad \ldots 4.2$$

It can be seen from the above equations that if $X(w_i)$ is very small, the effect of the noise term on the deconvolution result may be very large if $Y(w_i)$ is also

small. In fact, if one considers the noise level to be constant then when the value of $Y(w_i)$ is small, the effect of the noise on the deconvolution result becomes more prominent than the desired signal.

This deconvolution problem has been tackled in other areas such as astronomy where the effects of the telescope transfer function are removed from the observed signals. A fundamental difference between the deconvolution of the autocorrelation and crosscorrelation functions and the deconvolution of waveforms observed in astronomy is that the telescope transfer function is well defined, usually a smooth function which can be represented analytically and in most cases has no transmission zeros. If one equates the function X(w) as discussed previously with this telescope transfer function, we see that equation 4.2 converges for all $X(w_i)$ non zero. But if one allows $X(w_i)$ to tend towards zero for some W_i, then equation 4.2 may diverge. The results given earlier in this section display these divergent properties.

Deconvolution of two waveforms with an increasing amount of noise added to the output y(t) was carried out. The results given in Appendix 6 show that the deconvolution result is very sensitive to small amounts of added noise.

4.6 SUMMARY

The deconvolution program has been used to analyse simulated and experimentally measured correlation functions. The best deconvolution results have been those calculated from the simulated correlation functions. Successive deconvolution of two functions with increasing amounts of added noise have shown that this deconvolution technique yields very poor results when applied to real waveforms in the presence of noise. The application of simple mathematics has substantiated the sensitivity of the deconvolution result to noise inherent in the waveforms of interest.

It therefore appears that straightforward deconvolution has limited application and when deconvolving noisy waveforms, some process which uses an averaging technique to reduce the noise effects is desirable. One such technique known as Bayesian deconvolution is discussed in the following chapter.

CHAPTER 5

BAYESIAN DECONVOLUTION

5.1 INTRODUCTION

Investigation into other deconvolution techniques arising due to the failure of the method discussed in Chapters Two and Four has yielded the idea of Bayesian deconvolution. A brief treatment is given here with some preliminary results which are shown to be very promising.

Richardson, in Reference 5.1, has developed the idea of applying probability theory to the restoration of noisy, degraded spectra as found in spectroscopy. Using Bayes' theorem (Reference 5.2), Richardson has formulated an expression for the iterative estimation of an original image given the degraded image and the point spread function of the measuring system. Using matrix notation the iterative estimation may be described by:

$$T_i^{(n+1)} = T_i^{(n)} \left\{ \sum_k \frac{R_{k,i} \; M_k}{\sum_j R_{k,j} \; T_j^{(n)}} \right\} \qquad \ldots 5.1$$

where R is the response matrix, M the measured spectrum, T the true spectrum and n the iteration index.

Kennet et al. in Reference 5.3 have applied Bayesian deconvolution to photoneutron, x-ray and optical

spectroscopy amongst others. It appears that this method of deconvolution may readily be applied to the calculation of the transit time distribution by deconvolution of the autocorrelation and crosscorrelation functions as found in crosscorrelation flowmetering.

The parameters of equation 5.1 need to be changed to suit this slightly different application. Assuming y to be the system output vector, g the impulse response vector and x the input vector, we may rewrite equation 5.1 thus:

$$g_i^{(n+1)} = g_i^{(n)} + \sum_k \frac{X_{k,i}\, Y_k}{\sum_j X_{k,j}\, g_j^{(n)}}$$

Assuming s and p to be working vectors, the implementation of this expression as used by Kennet et al. may be described by the following five steps:

i. g = y
ii. p = x*g
iii. s = Y/P
iv. p = x ⊕ S
v. g = g.p

Where * and ⊕ denote the convolution and correlation operations respectively. The above algorithm is easier to implement using the speed and convenience of the fast Fourier transform. Using upper case letters for the respective Fourier transforms and representing the

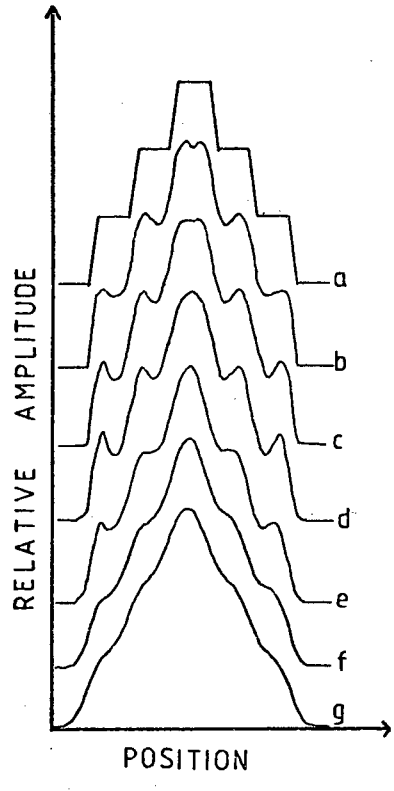
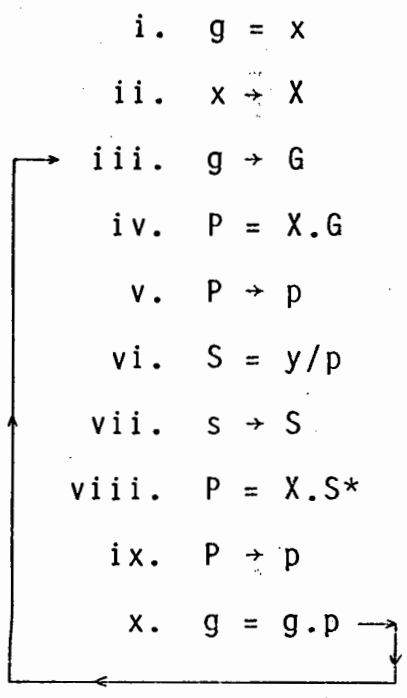


Figure 5.1 The deconvolution of a smeared step function and and a Gaussian function. Figure a shows the original step function, figure g the smeared step function and figures f to b the estimates after 2, 8, 32, 128 and 512 iterations respectively.

transformation operation via the Fourier transforms by an arrow, this algorithm may be rewritten:



The accuracy of the estimate of g improves as the number of iterations increases.

5.2 IMPLEMENTATION

The five step algorithm given above was incorporated as another, single instruction of the WAVEPACK program. The actual code required is only about twenty lines. The more efficient, ten step algorithm would be suited to a microprocessor implementation of this deconvolution technique.

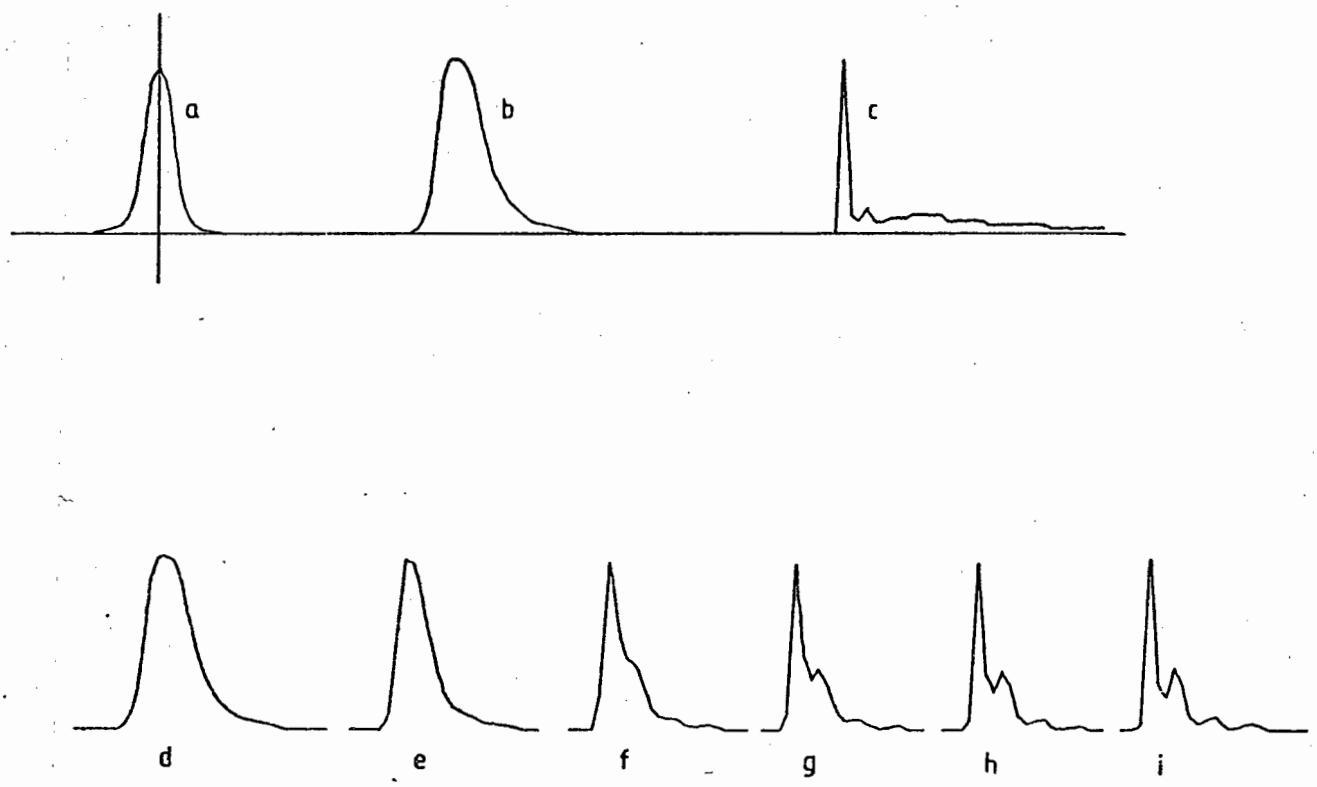


Figure 5.2 The deconvolution of the measured ACF and CCF from the experimental flow rig. The measured ACF and CCF are shown in a and b respectively, the transit time distribution obtained from the velocity profile measurement is given in figure c above. The convergence of the deconvolution is shown in d to i where the number of iterations is 0, 10, 50, 100, 150 and 200 respectively.

5.3 RESULTS

In order to validate the code written, similar tests to those carried out in Reference 5.2 were undertaken. A Gaussian function was convolved with the stepped function (given in Figure 5.1a) to produce a smeared step function (shown in Figure 5.1g). The deconvolution of this smeared step and the Gaussian functions was then carried out with an increasing number of iterations. The results are given in Figure 5.1 and they resemble very closely those obtained by Kennet et al.

The following step is obviously to apply this deconvolution technique to the autocorrelation and crosscorrelation functions measured experimentally and to compare the result with the measured transit time distribution. This has been done and the results are shown in Figure 5.2. The deconvolution produces a good approximation of the transit time distribution which improves as the number of iterations is increased. The variance between the measured and calculated transit time distributions falls fairly rapidly as the number of iterations is increased. This is shown in Figure 5.3.

5.4 CONCLUSION

The Bayesian deconvolution algorithm has been successfully applied to the deconvolution of a smeared step

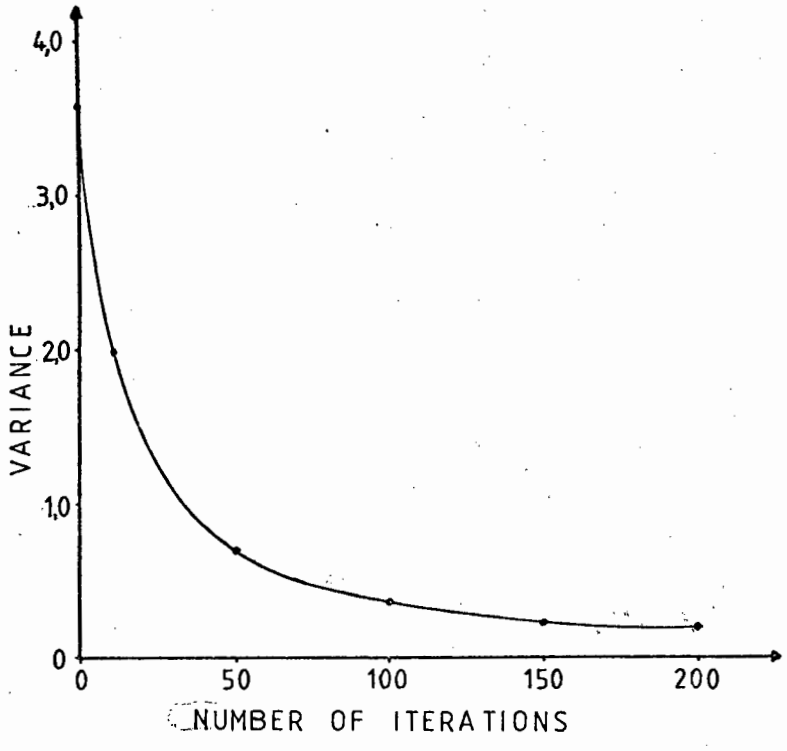


Figure 5.3 The variance between the measured and the calculated transit time distribution as a function of the number of iterations.

function and subsequently to the deconvolution of the autocorrelation and crosscorrelation functions as measured from the experimental flow rig described in Section 3.2. It remains now to propose possible implementations of this algorithm and to put forward further research ideas.

REFERENCES - CHAPTER 5

5.1 Richardson, W.H.: Bayesian-Based Iterative Method of Image Restoration. Journal of the Optical Society of America, Vol. 62, No. 1, pp.55-59.

5.2 Richmond, S.B.: Statistical Analysis. The Ronald Press Company, New York, 1964, pp.272-277.

5.3 Kennet et al.: Bayesian Deconvolution I, II and III Nuclear Instruments and Methods, Vol. 151 pp.285-292; Vol. 151, pp.293-301; Vol. 153, pp.125-135, 1978.

CHAPTER 6

DISCUSSION, RECOMMENDATIONS AND CONCLUSIONS

6.1 SUMMARY OF RESULTS

The aim of this project has been to investigate a technique to improve the range of operation and the accuracy of the crosscorrelation flowmeter. Efforts have been directed towards the calculation of the velocity distribution using signal processing techniques to analyse the autocorrelation and crosscorrelation functions. Errors introduced by the variation of the velocity profile in the flow being measured may be eliminated if the velocity distribution can be calculated from the flowmeter outputs.

Initially it was hoped to simulate the flow mechanism and the flow signal generation so that different correlation functions in the presence of varicus velocity profiles may be obtained. The simulation of the flow using a computer model appeared very convenient because the ease with which the different parameters may be varied. Two programs were envisaged, a simulation program in which a particular flow would be simulated followed by an analysis program to operate on the simulated flow signals in order to arrive back at the velocity distribution of the flow system.

For accurate simulation of the flow, the computer program was found to require excessive and impracticable amounts of computer time despite the various attempts at optimising the code. This led to the construction of a test rig so that the correlation functions and the prevailing velocity profile could be measured. These measured functions were then analysed as before. The flow rig proved to be very useful and the measurement of the velocity profile by crossed-beam correlation yielded a good result.

The analysis package was developed so that the deconvolution of the autocorrelation and crosscorrelation functions could be carried out. Many digital signal processing functions have been incorporated into this package. The deconvolution of simulated and experimentally measured correlation functions was carried out using the method of the division of the Fourier transforms. This technique has proved to be very sensitive to noise and the transit time distributions thus calculated bear no resemblance to the expected transit time distributions.

Experiments using a tapped delay line were carried out and the results have shown the noise present in the correlation functions to give rise to transmission zeros. A theoretical analysis has substantiated this explanation of these divergent results.

The failure of this direct method of deconvolution has led to a brief investigation into Bayesian deconvolution. The deconvolution of a stepped function and a Gaussian was carried out successfully using this algorithm. The deconvolution of the correlation functions measured experimentally agree well with the transit time distribution measured from the flow rig. The variance of the estimate is seen to reduce to 2% of the transit time distribution when calculated over 200 iterations. Implementation of this Bayesian deconvolution was achieved by adding it as a function carried out by the waveform analysis package.

The results given in this work detail only the investigation of a suitable algorithm which can be used to deconvolve the autocorrelation and crosscorrelation functions. No work has yet been carried out to implement this deconvolution as part of a crosscorrelation flowmeter.

6.2 FURTHER WORK

When using the crosscorrelation flowmeter, inaccuracies are introduced due to the presence of a curved velocity profile in the flow being measured. The effect of the velocity profile is to make the crosscorrelation function asymmetrical. If the requirement is to measure the mean transit time of the flow, then the

simplest strategy is to compute the transit time distribution using for example Bayesian deconvolution and then to calculate from this the average transit time. The average velocity may then be calculated from the average transit time. One can envisage a modern generation, sixteen bit microprocessor sampling the two flow signals, calculating their autocorrelation and crosscorrelation functions, then applying Bayesian deconvolution to calculate the transit time distribution and from this computing the mean transit time and hence the average velocity. The calculation of the correlation functions and the implementation of the Bayesian deconvolution algorithm would require the implementation of a fast Fourier transform using a microprocessor. This has already been shown to be realistic even though a dedicated unit may be required.

The correlation functions calculated using a polarity correlator are related to the direct correlation function by the Van Vleck equation, that is:

$$R_D = \sin\left(\frac{\pi}{2} R_p\right)$$

where R_D is the normalised, direct correlation function and R_p the polarity correlation function. Jordan (Reference 6.1) suggests a method for simplifying the calculation of the discrete Fourier transform of a correlation function found using a polarity correlator.

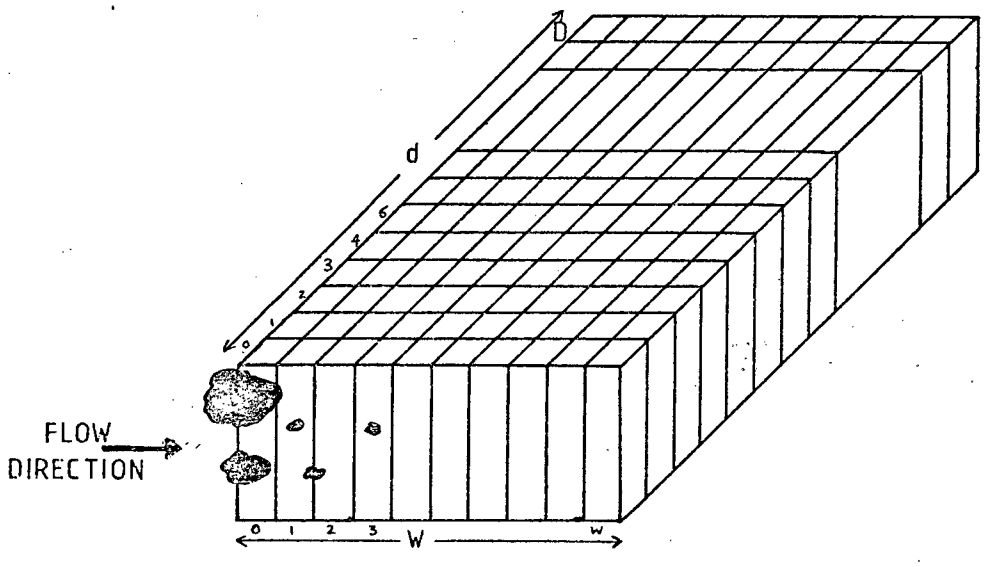


Figure 6.1 Movement of particles in a channel of width d through the beam quantized into W smaller beams.

He uses the above equation to reduce the calculation of the discrete Fourier transform to trigonometric and addition functions in much the same way as the fast Fourier transform is developed. This idea may be applied to the calculation of the transit time distribution via the Bayesian deconvolution algorithm.

Further research may be aimed at the actual realisation of this crosscorrelation flowmeter which incorporates all the above innovations.

Rametti (in Reference 6.2) has proposed the notion of identifying the flow system using only a single measuring station. If one considers this single transducer to produce an interrogating beam which is wider than the largest particle in the flow, then the process of the flow signal generation may be modelled as the measurement of the flow particles through a quantized beam. This is shown schematically in Figure 6.1 where there are W smaller beams. This proposal assumes a rectangular beam cross-section although this is not necessary. The approach adopted here is similar to the operation of the parallel-slit image velocity sensor as discussed in Reference 6.3.

Consider now a fluid element representing a fraction b_i of the pipe total cross-sectional area, which proves at a velocity V_i (as shown in Figure 6.2). This will

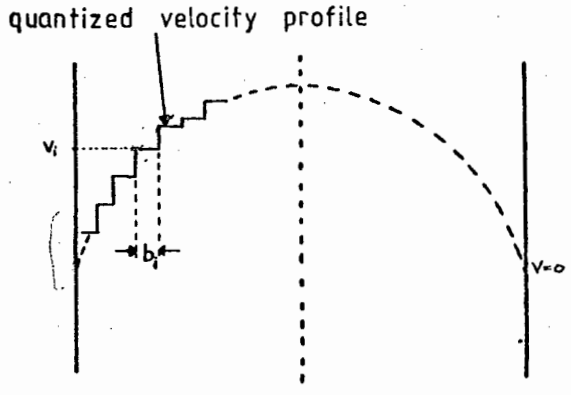


Figure 6.2 Cross-sectional view through pipe showing the quantized velocity profile.

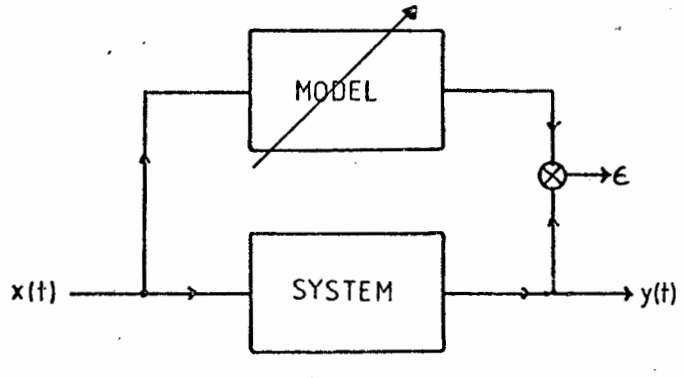


Figure 6.3 Identification of the system by modelling the system parameters.

cause a given disturbance in the i^{th} transducer beam after a delay of T_i sampling intervals. The transport mechanism may be expressed mathematically as:

$$y(k) = b_0 u\ (k-T_0) + b_1\ U\ (k - T_1) + \ldots$$

$$\ldots + b_W\ U\ (k-T_W)$$

$$= B(Z^{-1})\ U\ (k)$$

Well accepted methods for computing $B(Z^{-1})$ are outlined in Reference 6.4 although in this case the flow system is of simpler form than those in the reference. In general terms the system to be identified is modelled and an error function formed from the calculated and the measured system outputs is minimised by suitable variation of the model parameters (see Figure 6.3).

The system identification program developed by Mr. L.B. Rametti was run successfully with flow signals simulated with a flat velocity profile as the system inputs and outputs. This program optimises the model parameters using a least squares error minimisation algorithm. Unfortunately, major changes are required to the existing software to cater for the unusually long delays which characterise the flow system.

This system identification approach should be investigated

to determine the feasibility of applying this least squares error minimisation to an on-line flow system identification scheme.

6.3 CONCLUSIONS

It has been the aim of this work to investigate a method for calculating, from the outputs of a cross-correlation flowmeter, the velocity distribution in the flow being measured. The need to measure the velocity distribution arises from the efforts to increase the accuracy and range of the crosscorrelation flowmeter. Measurement of the flow velocity in the presence of complex velocity profiles gives rise to error in the estimation of the flow transit time. The approach adopted here has been to consider the flow and transducer as a system with the flow signals as inputs and outputs and to then identify this system to obtain the transit time distribution.

Several deconvolution methods have been outlined in Chapter 2 and one of these examined in detail. Due to the extreme sensitivity of this first approach to noise, a second deconvolution method has been examined. The results of this later investigation appear to overshadow the work done in applying the simpler deconvolution method. Experiments have shown that the presence of

additive noise led to the failure of this otherwise satisfactory deconvolution algorithm. Valuable experience in the application and theory of digital signal processing has been gained and this work is felt to have been worthwhile. This alternative method of Bayesian deconvolution has been briefly examined with good results which show this technique to be directly applicable to the calculation of the transit time distribution.

Some of the groundwork for the application of digital signal processing techniques to crosscorrelation flowmetering has been carried out. The techniques discussed here have definite application to the improvement of range and accuracy of the existing flowmeter. The implementation of these methods to the measurement of the mean flow velocity in the presence of complex velocity profiles has still to be carried out.

It is hoped that the work presented here will provide a contribution to the field of crosscorrelation flowmetering and electrical engineering in general.

REFERENCES - CHAPTER 6

6.1 Jordan, J.R.: Discrete Fourier Transforms of Correlation Functions. Electronics Letters, Vol. 15, No. 6, 15 March 1979, pp.196-197.

6.2 Rametti, L.B.: Private Communication. University of Cape Town, May 1981.

6.3 Ator, J.T.: Image Velocity Sensing by Optical Correlation. Applied Optics, Vol. 5, No. 8, August 1966, pp.1325-1331.

6.4 Åström, K.J. and Eykhoff, P.: System Identification - A Survey. Automatica, Vol. 7, January 1971, pp.123-162.

APPENDIX 1

A SIGNAL PROCESSING MICROPROCESSOR

It is intended to develop a microprocessor based, crosscorrelation flowmeter at some later stage incorporating into it a correction for the velocity profile in the flow being measured. This microprocessor system was developed as a building block for the flowmeter.

Basically it consists of five parts, a standard Intel SDK85, extended RAM and EPROM facilities, two analogue to digital converters (ADCs), an interrupt clock and a display unit. The reason for the two ADCs was to make it possible to use the microprocessor system to perform the correlation of the flow signals.

The SDK85 is well known and needs no explanation.

The two analogue to digital converters are 8 bit, bus compatible ADC0804s. These have a maximum conversion time of 100 μsecs which is fairly slow by normal standards but adequate for this application. The ADCs are memory mapped and to start conversion the desired ADC is written to. At the end of conversion the corresponding bit of PORT 1 of the SDK85 is set low. The ADC is reset when the 8 bit value is read from

it. ADC 1 is addressed at 9800 to 9FFF and ADC2 at A000 to A7FF.

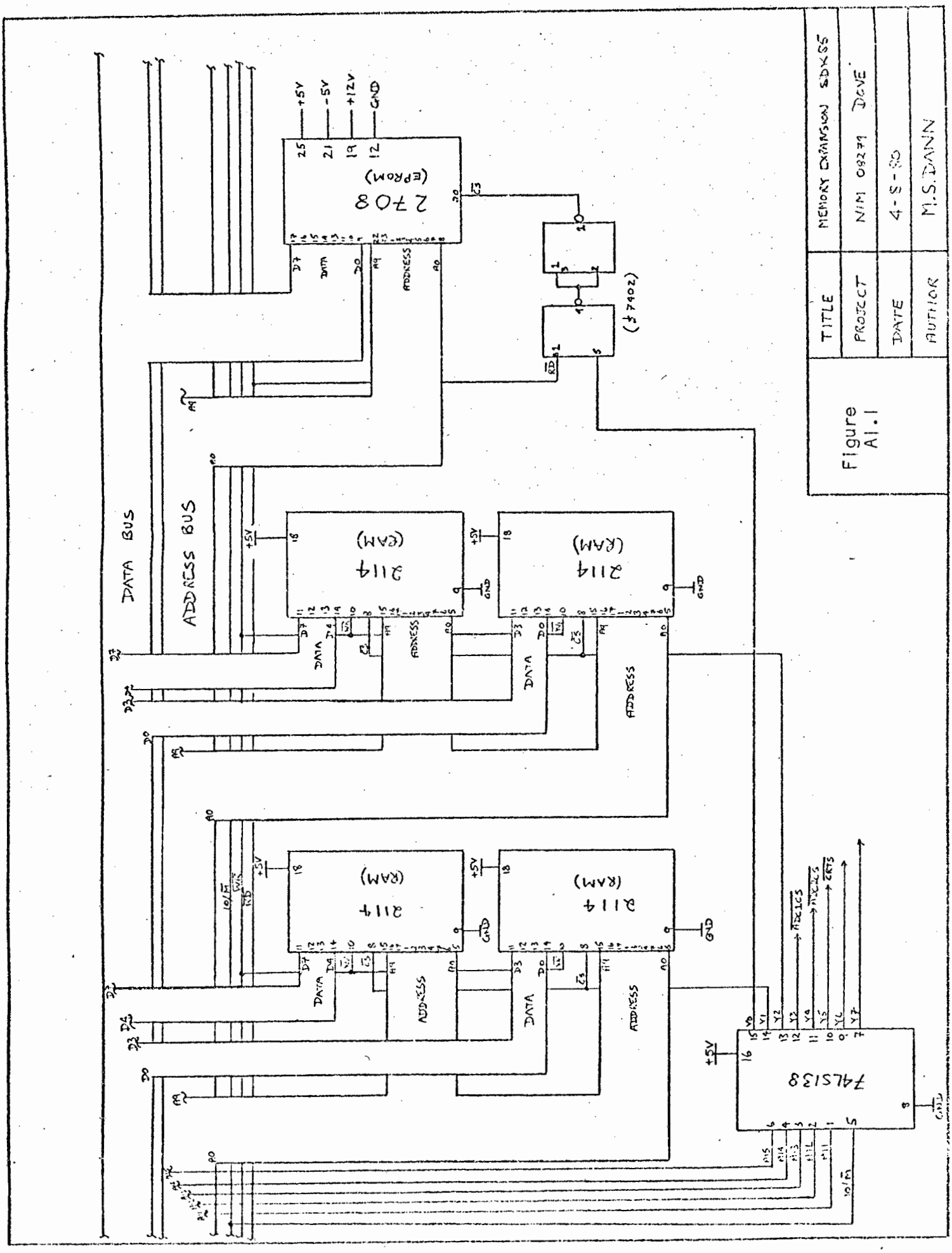
The interrupt clock operation is very simple; a 12 bit counter which has one of its outputs switch selectable. The selected counter output is then used to clock a '1' into a D type flip flop whose output is connected to the RST6.5 interrupt line of the SDK85. It is necessary to use the flip flop because the RST6.5 interrupt is level sensitive. It is necessary to clear the flip flop having completed the RST6.5 interrupt service routine, which is done by sending a '0' to bit 5 of PORT 1.

The display section allows 1K of memory to be displayed on an oscilloscope screen independently of the central processing unit. The display RAM is memory mapped but may only be written to or read from when it has been selected by the CPU. To select the display RAM bit 4 of PORT 1 is set to a '1'. This has the effect of blanking the display, using the oscilloscopes Z-mod input, and switching the address and data lines of the CPU to the display RAM. When deselected, i.e. bit 4 of PORT 1 is set to '0', each location of the display RAM is addressed sequentially using a counter. The data contained in these memory locations is then converted to an analogue voltage using a digital to analogue converter (DAC). Originally it was intended to use

the oscilloscope in the X-Y mode, thus a voltage ramp is provided by another DAC which simply uses the RAM address lines as its input. It has proved to be less problematic to use the ramp as an external trigger so the DAC which generates this ramp is not really needed, a single pulse would be sufficient to trigger the timebase.

This display RAM is addressed from A800 to AFFF (with a 1K foldback). The oscilloscope trace is blanked while the RAM address is changing. This is done using a monostable which is triggered by the lowest order address line which modulates the Z-mod input to the oscilloscope (shown in Figure A1.4).

Examples of a sampled waveform, its Fourier transform and inverse Fourier transform is shown in Figure A1.5. These are taken from the work done by Weeks in Reference 2.12.


MEMORY EXPANSION 8DKSS
NIM 09239 DAVE
4-5-80
M.S.DANN
TITLE
PROJECT
DATE
AUTHOR
Figure A1.1
2708 (EPROM)
2114 (RAM)
74LS138
DATA BUS
ADDRESS BUS
+5V
-5V
+12V
GND

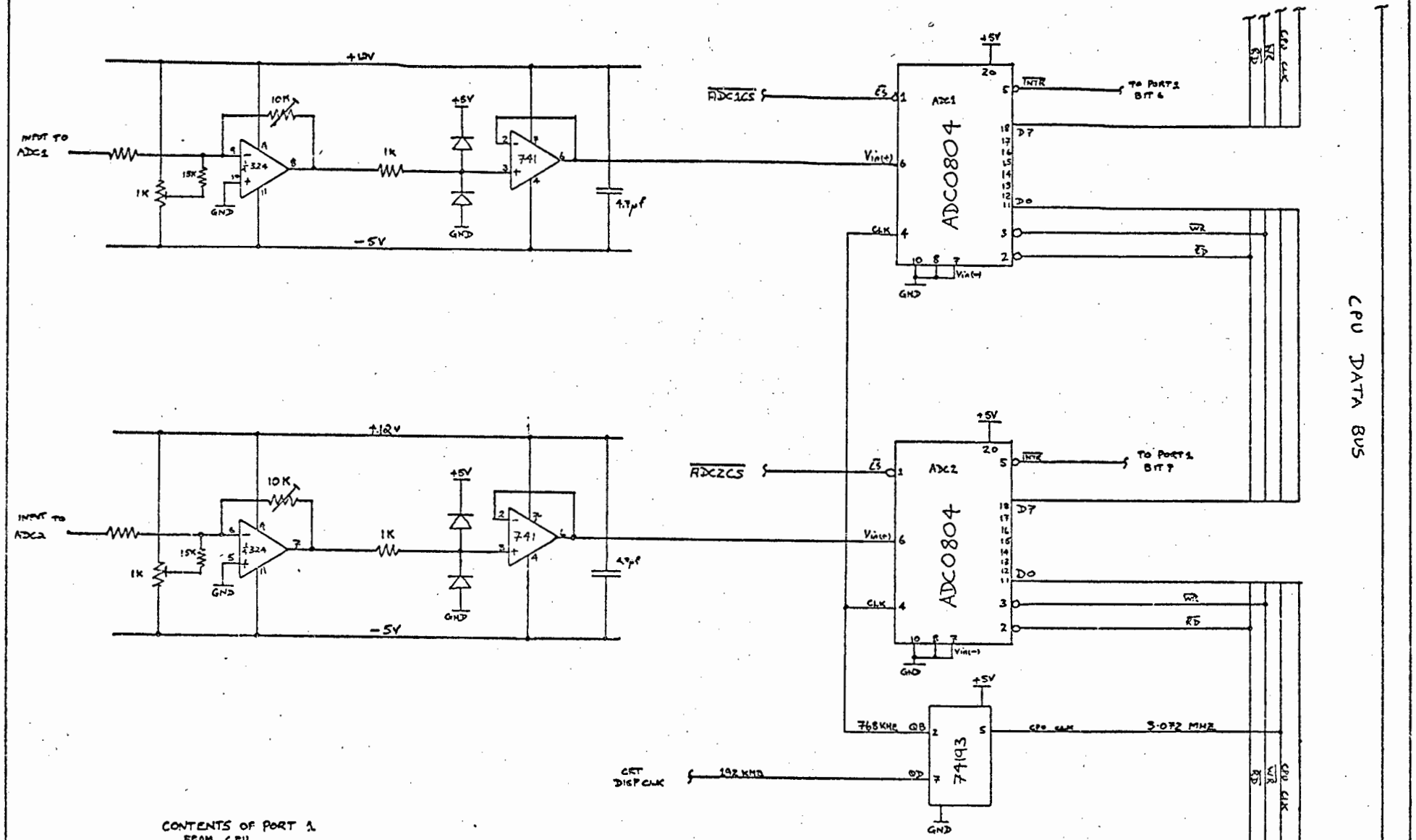
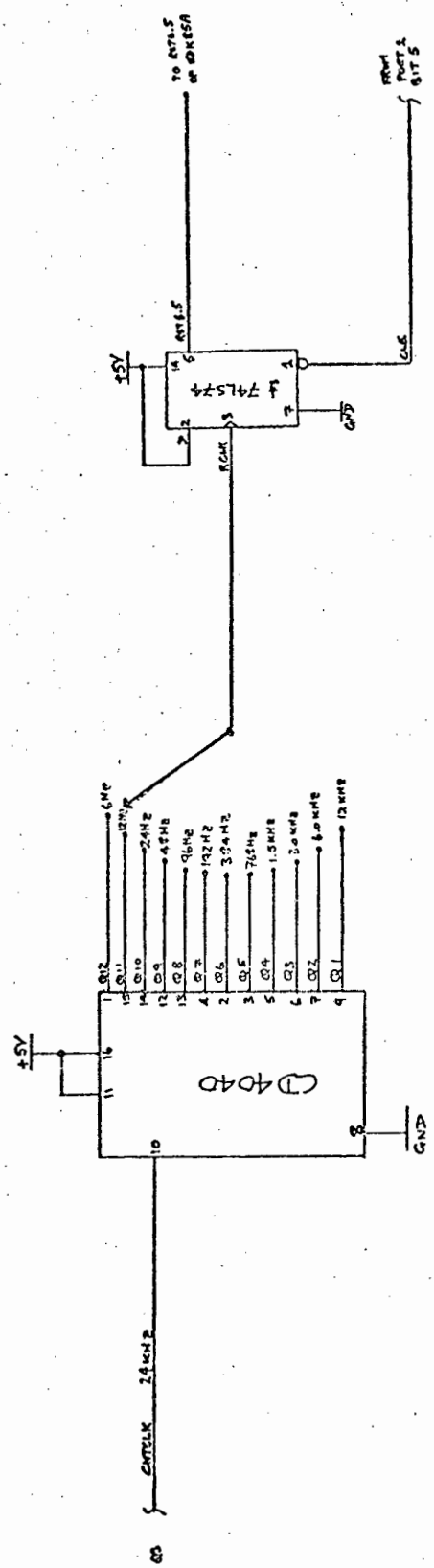


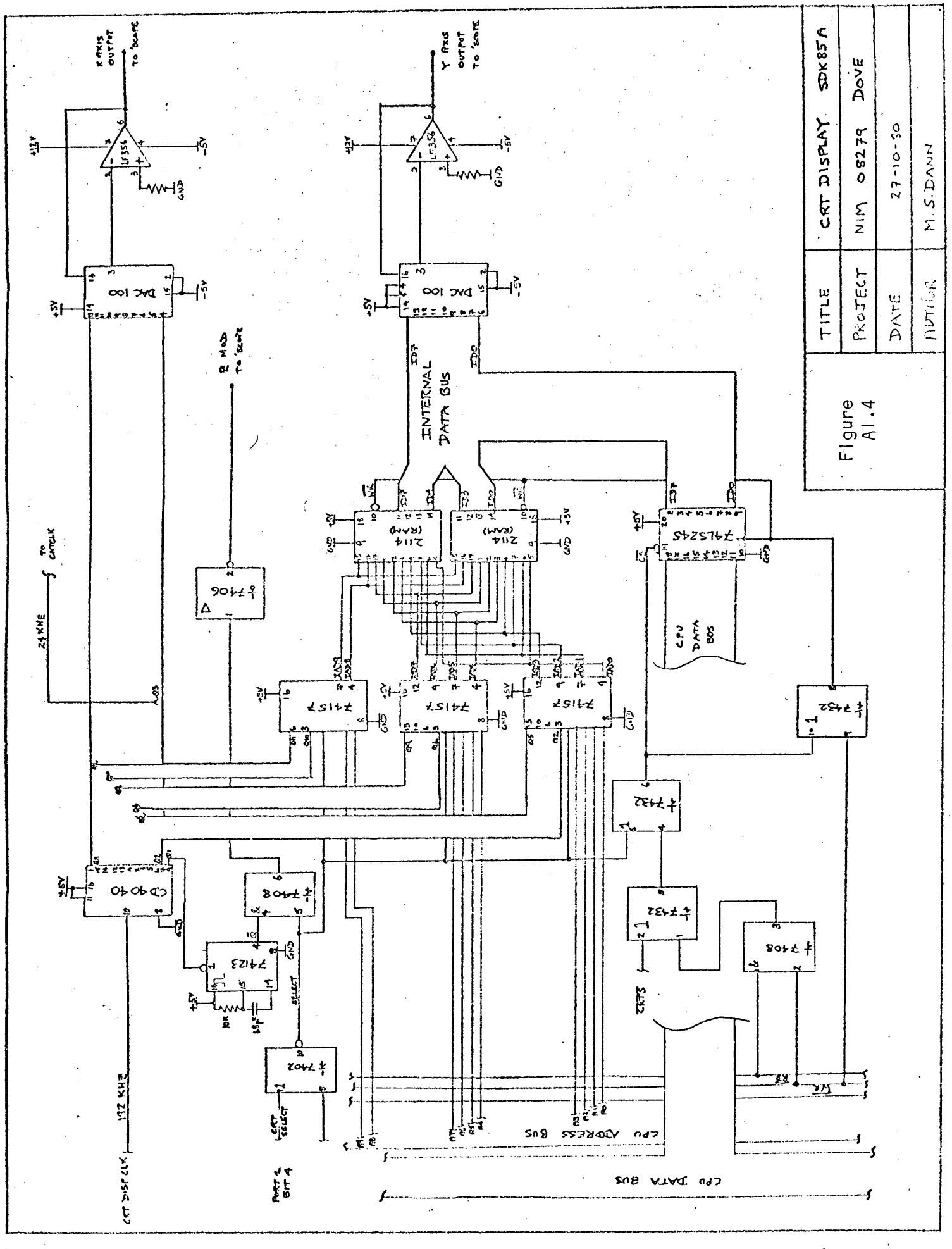
Figure A1.2

TITLE	A-D CONVERTERS SDK85A
PROJECT	NIM 08279 DOVE
DATE	18-8-80
AUTHOR	M. S. DANN



TITLE	INTERRUPT CLOCK SDK 85A
PROJECT	NIM 08279 DOVE
DATE	28-10-80
AUTHOR	M.S.DANN

Figure A1.3


TITLE
CRT DISPLAY SDK85A
PROJECT
NIM 08279 DOVE
DATE
27-10-30
AUTHOR
M.S.DANN
Figure A1.4
X AXIS OUTPUT TO SCOPE
Y AXIS OUTPUT TO SCOPE
Z MOD TO SCOPE
INTERNAL DATA BUS
CPU DATA BUS
CPU ADDRESS BUS
CRTS
TO CAMCLK
24 KHz
192 KHz
CRT DISP CLK
PORT 2 BIT 4
CRT SELECT
SELECT
DAC 100
2114 (RAM)
2114 (RAM)
74LS245
74157
74157
74157
74LS32
74LS32
74LS32
74LS08
74LS08
74LS02
74LS123
CD4040
74LS06

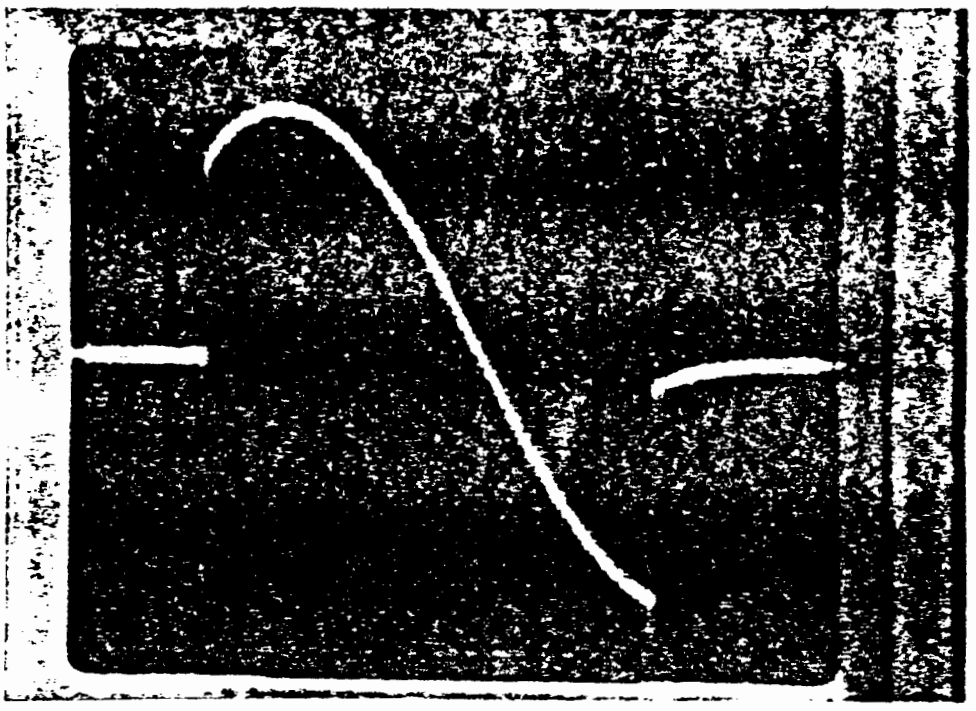


Figure A1.5a Sampled portion of a sine-wave.

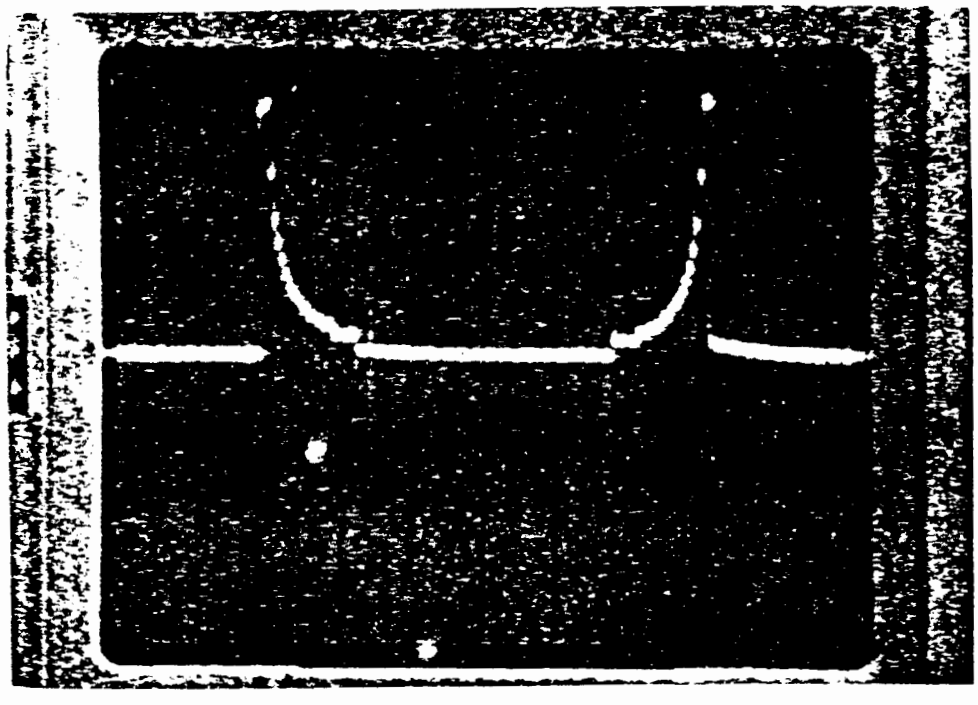


Figure A1.5b FFT magnitude function showing leakage due to the sharp discontinuity of the data.

APPENDIX 2

SIMULA V2.0

This program is designed to simulate multiphase fluid flow in a pipe and generate flow signals. The program outputs are the autocorrelation and crosscorrelation functions of the upstream and downstream signals.

The code is well documented and fairly self-explanatory. Various parameters which are of interest are: NPTS which specifies the number of data points desired in the calculation of the autocorrelation and crosscorrelation functions; LENGTH and WIDTH which determine the desired pipe length and width; AVLNGT specifies the transducer beamwidth and ACFILE and CCFILE set up the desired output file numbers.

The program inputs are (i) the number of times the autocorrelation and crosscorrelation functions are to be averaged, and (ii) the desired velocity profile consisting of integers between zero and nine. A typical runstream and the subsequent program printout is shown overleaf.

In this example SIMULA writes the autocorrelation and crosscorrelation functions to the data files ACF100-V2 and CCF100-V2 respectively. The program prints out

```
@RUN CAN1,A0510-R115/MD101699T,DOVEMD,10,50
@ASG,A ACF100-V2.
@ASG,A CCF100-V2.
@USE 11.,ACF100-V2.
@USE 12.,CCF100-V2.
@ASG,A PROJECT.
@XQT PROJECT.SIMULA
100
2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
@FIN

SIMULA       V2.0      17/02/81


PIPE LENGTH =  50     PIPE WIDTH =  20

NUMBER OF SAMPLES =   64     AVERAGING WIDTH =  10

NUMBER OF RECORDS AVERAGED =    100

VELOCITY PROFILE USED IS :

  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2

ACF NORMALIZING FACTOR =       7.7483     CCF NORMALIZING FACTOR =      5.1628
```

Figure A2.1 The runstream and corresponding program output.

the parameters used and the normalising factors, as is seen here. Subsequent processing, plotting or printing is done using the WAVEPACK program which is detailed in the following Appendix. Further examples are given in Chapter 4 and Appendix 5, where the simulated correlation functions are analysed.

The autocorrelation and crosscorrelation functions resulting from this runstream are given in Figure A2.2.

Autocorrelation and crosscorrelation functions found from the simulated flow signals computed using more complex velocity profiles are shown in Figures A2.3, A2.4 and A2.5. All these correlation functions have been computed by averaging over 100 flow signal samples.

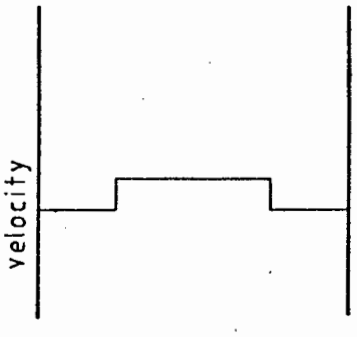


Figure A2.5a The stepped velocity profile.

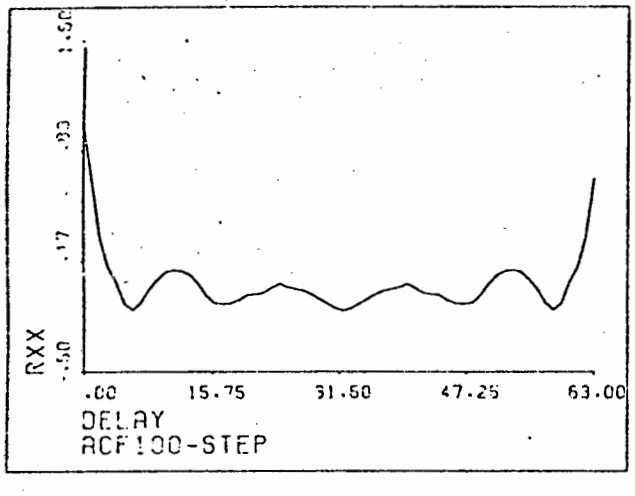


Figure A2.5b The autocorrelation function.

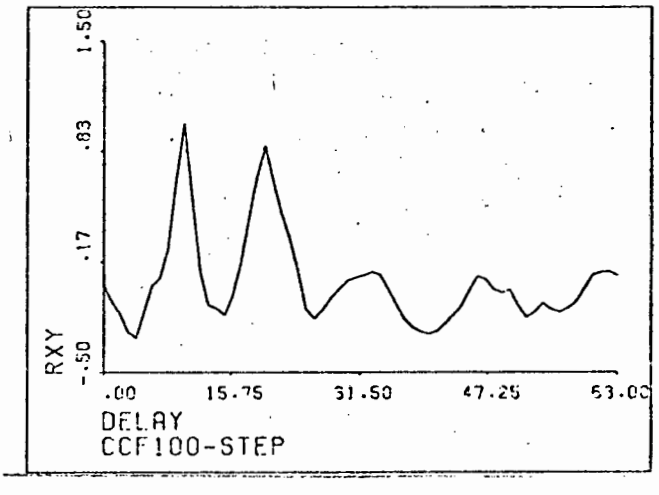


Figure A2.5c The crosscorrelation function.

```
C  TITLE :  SIMULA
C
C
C  DESCRIPTION :
C
C This program is designed to simulate multiphase fluid flow in a pipe
C and to generate flow signals. The program outputs are the  auto  and
C crosscorrelation functions of the upstream  and  downstream signals.
C
C
C  CALLS  : RANDN
C
C
C  CODE :
C
C
C  Definition of variables to be used.
C ====================================
C
      IMPLICIT COMPLEX  (A - Z)
C
      PARAMETER  NPTS = 64
      PARAMETER  LENGTH = 50, WIDTH = 20
      PARAMETER  AVLNGT = 10
      PARAMETER  ACFILE = 11, CCFILE = 12
C
      REAL  UPSIG(0:NPTS-1), DWNSIG(0:NPTS-1), ACF(0:NPTS-1)
      REAL  CCF(0:NPTS-1)
      REAL  DATA(1:WIDTH, 1:LENGTH), RNDARY(1:WIDTH), TMPRND(1)
      REAL  MAXACF, MAXCCF, NOSUMS
C
      INTEGER  TEMP, INDEX, NOAVE, WHERE, SHIFT
      INTEGER  I, J, K, L
      INTEGER  VPROF(1:WIDTH)
C
C
C    Printing heading.
C   ==================
C
      PRINT 100
C
C
C    Reading the number of records the acf and the ccf are to be
C    averaged over and the desired velocity profile.
C   ================================================================
C
      READ *, NOAVE
      READ *, ( VPROF(I), I=1,WIDTH )
C
C
C    Printing out the selected parameters.
C   ======================================
C
      PRINT 20, LENGTH, WIDTH, NPTS, AVLNGT
20    FORMAT(///,3X,'PIPE LENGTH = ',I3,5X,'PIPE WIDTH = ',I3,3X,
     &        //,3X,'NUMBER OF SAMPLES = ',I4,5X,'AVERAGING WIDTH = ',
     &        I3)
      PRINT 30, NOAVE,(VPROF(I), I = 1,WIDTH)
30    FORMAT(//,3X,'NUMBER OF RECORDS AVERAGED = ',I4,//,3X,
     &        'VELOCITY PROFILE USED IS : ',//,3X,20(2X,I2) )
C
C
C
C    Initializing variables.
C   ========================
C
      TMPRND(1) = 314159
      RNDARY(1) = 236097
      NOSUMS = FLOAT( AVLNGT * WIDTH )
C
C
C    Initializing fluid elements in the pipe.
C   =========================================
C
      DO 1000  I = 1,LENGTH
C
         RNDARY(1) = RNDARY(1) * 100000000.
         CALL  RANDN( RNDARY, WIDTH, 0., .5 )
C
         DO 1010  J = 1,WIDTH
            DATA(J,I) = RNDARY(J)
1010     CONTINUE
C
1000  CONTINUE
C
```

```
C
C    Computing the flow signals, ACF and CCF for each record.
C    ========================================================
C
      DO 2000  I = 1,NOAVE
C
C
C     Repeat the following section for each sample of the flow signals.
C     =================================================================
C
         DO 3000  INDEX = 0,NPTS-1
C
C
C        Implementing velocity profile effects for each sample.
C        ======================================================
C
            DO 3100  J = 1,WIDTH
C
               TEMP = VPROF(J)
C
               IF  ( TEMP .NE. 0 )  THEN
C
                  DO 3110  K = 1,TEMP
C
                     DO 3120  L = LENGTH,2,-1
                        DATA(J,L) = DATA(J,L-1)
3120                 CONTINUE
C
C
                     TMPRND(1) = TMPRND(1) * 100000000.
                     CALL  RANDN( TMPRND, 1, 0., 0.5 )
                     DATA(J,1) = TMPRND(1)
C
3110              CONTINUE
C
               ENDIF
C
3100        CONTINUE
C
C
C     Calculating the flow signal samples.
C     ====================================
C
         DO 3200  J = 1,AVLNGT
            WHERE = LENGTH - AVLNGT + J
C
            DO 3210  K = 1,WIDTH
               UPSIG(INDEX) = UPSIG(INDEX) + DATA(K,J)
               DWNSIG(INDEX) = DWNSIG(INDEX) + DATA(K,WHERE)
3210        CONTINUE
C
3200     CONTINUE
C
C
C    Normalizing the flow signals.
C    =============================
C
         UPSIG(INDEX) = UPSIG(INDEX) / NOSUMS
         DWNSIG(INDEX) = DWNSIG(INDEX) / NOSUMS
C
C
3000     CONTINUE
C
C
C
C    Calculating the ACF and the CCF from the flow signals.
C    ======================================================
C
         DO 4000  J = 0,NPTS-1
C
            DO 4010  K = 0,NPTS-1
               SHIFT = MOD(J+K,NPTS)
               ACF(J) = ACF(J) + UPSIG(K) * UPSIG( SHIFT )
               CCF(J) = CCF(J) + UPSIG(K) * DWNSIG( SHIFT )
4010        CONTINUE
C
4000     CONTINUE
C
C
         DO 10  J = 0,NPTS-1
            UPSIG(J)=0.
            DWNSIG(J)=0.
10       CONTINUE
C
C
2000  CONTINUE
C
C
```

```
C
C
C    Normalizing the averaged ACF and CCF.
C   ======================================
C
      MAXACF = ABS( ACF(0) )
      MAXCCF = ABS( CCF(0) )
C
      DO 5000  I = 1,NPTS-1
         IF ( ABS( ACF(I) ) .GT. MAXACF )  MAXACF = ABS( ACF(I) )
         IF ( ABS( CCF(I) ) .GT. MAXCCF )  MAXCCF = ABS( CCF(I) )
5000  CONTINUE
C
      DO 6000  I = 0,NPTS-1
         ACF(I) = ACF(I) / MAXACF
         CCF(I) = CCF(I) / MAXCCF
6000  CONTINUE
C
C
C    Printing the values of ACFMAX and CCFMAX.
C   ==========================================
C
      PRINT 300, MAXACF, MAXCCF
C
C
C
C
C    Writing the ACF and CCF data to a permanent file.
C   ==================================================
C
      DO 7000  I = 0,NPTS-1,8
         WRITE ( ACFILE, 200 ) ( ACF(I+J), J = 0,7 )
         WRITE ( CCFILE, 200 ) ( CCF(I+J), J = 0,7 )
7000  CONTINUE
C
C
      STOP
C
C
C    Format statements.
C   ===================
C
100   FORMAT ( 1H1, ///, 3X, 'SIMULA       V2.0      17/02/81 ')
C
200   FORMAT ( 8(3X,F12.6) )
C
300   FORMAT (//,3X,'ACF NORMALIZING FACTOR = ',F12.4,5X,
     &        'CCF NORMALIZING FACTOR = ',F12.4,///)
C
C
C
C
      END
END OF FILE
->
```

APPENDIX 3

WAVEPACK

A3.1 INTRODUCTION

This program has been designed to function as an interactive waveform analysis package. It is intended to be used as a tool to simplify the implementation of signal processing techniques to various practical problems. Motivation for the development of the program has been to aid the work done to deconvolve two waveforms using Fourier transform techniques. The software has been written in Ascii Fortran level 8R1 and implemented on a Univac 1106 multiprocessor, time-sharing system.

A3.2 SPECIFICATIONS

The program functions almost as a powerful pocket calculator with fairly complex operations on large quantities of data. Four work areas WA1, WA2, WA3 and WA4 are available to the user as scratchpad type memory. It is on these work areas that all the commands are carried out. To execute a command the user types in from the terminal, the command name followed by the various parameters required. One may consider each work area to be an array which contains a sampled waveform.

For example, these work areas may be used to plot graphs of data, convolve or deconvolve two waveforms. Many other functions which are available are described in detail below. The basic theory behind each of the signal processing commands is covered in Chapter 2 of the main text.

A3.3 PROGRAM DESCRIPTION

WAVEPACK calls several useful subroutines which are also listed here. They are:

1. FFT a subroutine which implements the fast Fourier transform.

2. INVFFT computes the inverse fast Fourier transform.

3. LINFLD allows the program to have flexible command formats.

4. GRAPHW and GRAPHV generate a character plot of the data in any work area.

5. PRINT lists the data in any work area.

6. WINDOW applies various data windows to any work area.

7. SLICIT operates as an interface for the Calcomp plotter so that accurate plots of the data in any work area may be generated.

8. NEWPAG and PAGNAM are subroutines required to interface to the Graphics Display Package and are system library commands (hence not listed here).

Specification of the number of data elements in the work areas or the number of temporary storage areas are declared in the code as parameters. These may be changed by the user if the need arises by simply changing the source code, recompiling and collecting. In the listing given here the number of data is 64 points per work area. The parameters are:

1. M and NPTS, where NPTS specifies the number of data elements required for each work area and M is simply found from the relation NPTS = 2^M.

2. NFLDS specifies the maximum number of input fields required for the WAVEPACK commands. Commas separate each field in a line entered by the user. If modifications are made and commands added which require more than 5 fields then this parameter should be changed accordingly.

3. CLNGTH is a parameter which specifies the maximum number of characters in any command.

4. FLNGTH specifies the maximum number of characters in each field.

5. NSTORS specifies the number of temporary storage areas available (see Figure A3.1).

6. NWAREA specifies the number of work areas available. Changes to to source code will, however, be required if the functions available are to be implemented using more than four work areas.

7. PFILE specifies the device number which will refer to the printfile.

The program allows the user to set up labels for each temporary storage area and each work area. These labels may then be listed so that each storage area can be easily identified. The labels which are printed for the graphs and data listings may have a maximum length of 12 characters.

A3.4 THE WAVEPACK COMMANDS

Notation:

XXX = WA1, WA2, WA3 or WA4

A = an integer $0 \leq A \leq NPTS - 1$

B = an integer $A \leq B \leq NPTS - 1$

N = an integer the value of which depends upon the particular command.

C,STDEV,AMPL
SEED = any real number

The Commands:

1.	a. HELP	Prints an index of the WAVEPACK commands.
	b. HELP,ALL	Prints all the WAVEPACK commands page by page.
	c. HELP,A	Prints the commands on page A.
2.	SAVE,N,XXX	Stores work area XXX permanently in Fortran file number N+10.
3.	RETN,N,XXX	Returns work area XXX from Fortran file number N+10.
4.	RCL,N,XXX	Recalls work area XXX from temporary store N.
6.	SLABEL,XXX,'label'	Sets the label of work area XXX.
7.	RLABEL,XXX	Prints the current label of work area XXX.
8.	SLABEL,N,'label'	Sets the label of temporary store N.
9.	RLABEL,N	Prints the current label of temporary store N.
10.	LLABEL	Lists all the current used defined labels.
11.	PRINT,XXX	Prints the data in work area XXX on the vdu screen.
12.	PRINTF,XXX	Sends the data in work area XXX to the print file.

13. GRAPH,XXX — Plots a character graph of work area XXX on the vdu screen.

14. GRAPHF,XXX — Sends a character graph of the data in work area XXX to the print-file.

15. FFT — Calculates the fast Fourier transform of a complex function using the data in work area 1 as the real part and the data in work area 2 as the imaginary part. The real and imaginary parts of the resulting Fourier transform are returned to work areas 1 and 2 respectively.

16. INVFFT — Computes the inverse fast Fourier transform using the work areas as for the FFT command above.

16. WINDOW,XXX — This command applies one of five windows to work area XXX. On execution a further prompt is issued for the user to type in the window code and window centre. The window codes are:

1 - zero order window
2 - first order window
3 - Hanning window
4 - Hamming window
5 - Blackman window

The window centre is specified by typing in an integer value of the desired x-axis centre point. The various windows and their Fourier transform are plotted in Figure A3.4 and an example of the use of this command is given in Section A3.6.

18. R-P

This command initiates a rectangular to polar conversion with work areas 1 and 2 as the real and imaginary parts on input which are replaced by the magnitude and angle respectively on output.

19. P-R

This initiates a polar to rectangular conversion with the data in work areas 1 and 2 as the magnitude and phase components on input which are replaced with the real and imaginary parts respectively as output.

20. CONV

This command implements the cyclic convolution of the data contained in work areas 1 and 2. The result is placed in work area 3.

21. a. DECONV1

This command is the first of three which implement the step by step deconvolution of two waveforms. Initially WA1 = Re(x), WA2 = Im(x), WA3 = Re(y) and WA4 = Im(y). After execution of DECONV1 we will have WA1 = Re(FT(x)), WA2 = Im(FT(x)), WA3 = Re(FT(y)) and WA4 = Im(FT(y)).

b. DECONV2

This initiates the second stage of the deconvolution by performing WA3 = Re(FT(y)/FT(x)) and WA4 = Im(FT(y)/FT(x)). (FT(y)/FT(x)).

c. DECONV3

This is the last stage of the deconvolution and implements WA1 = $Re(FT^{-1}\ FT(y)/FT(x))$ and WA2 = $Im(FT^{-1}\ FT(y)/FT(x))$.

22.	SUBST,XXX,A	Enters the substitute mode with work area XXX starting at data element A. This command enables the user to enter, change or delete any element of the work area. For an example of the operation of this command, see Section A3.6.
23.	FILL,XXX,C,A,B	This sets the data elements from A to B inclusive of work area XXX equal to C.
24.	NORM,XXX	Normalises all the data in work area XXX and prints the normalising factor.
25.	SINE,XXX	Performs the sine transform on work area XXX as given by the Van Vleck relation (see Section 3.2.3).
26.	ARCSINE,XXX	Performs the inverse sine transform (see above).
27.	ROTATE,XXX,A	Carries out the cyclic shifting of work area XXX moving each data element A times. Positive A denotes a right shift and negative A a left shift.
28.	XXX=YYY	Sets the data in any work area equal to the data in any other work area.
29.	SWA1=WA2,A,B	Sets the data elements A to B in work area 1 equal to the corresponding data elements in work area 2.
30.	SWA=WA1,A,B	See above.

31.	WA1=WA1*WA2	The following commands are self-explanatory
32.	WA1=WA1/WA2	
33.	WA1=WA2/WA1	
34.	WA1=WA1+WA2	
35.	WA1=WA1-WA2	
36.	WA1=WA2-WA1	
37.	WA1=1/WA1	
38.	WA1=WA1*,C	Multiplication of each element of work area 1 by the value C
39.	WA1=WA1/,C	Division of each element of work area 1 by the value C.
40.	WA1=WA1+,C	Addition of C to each element of work area 1.
41.	WA1=WA1-,C	Subtraction of C from each element of work area 1.
42.	NOISE+,XXX,STDEV, AMPL,SEED	Performs the addition of a pseudo-random noise sequence with a normally distributed amplitude to work area XXX. The standard deviation, amplitude and seed required are also entered.
43.	PLOT,XXX	This command allows the user to send the data in work area XXX to the Calcomp plotter. A detailed example illustrating the use of this command is given in Section A3.6.

44. BAYDECRS,N — This command implements the Bayesian deconvolution described in Chapter 5 of the main text. The number of iterations desired is specified by the value N. Work area 1 should contain the input waveform data and work area 2 the output waveform data. The resulting impulse response is returned in work area 3.

45. STOP — Terminates WAVEPACK

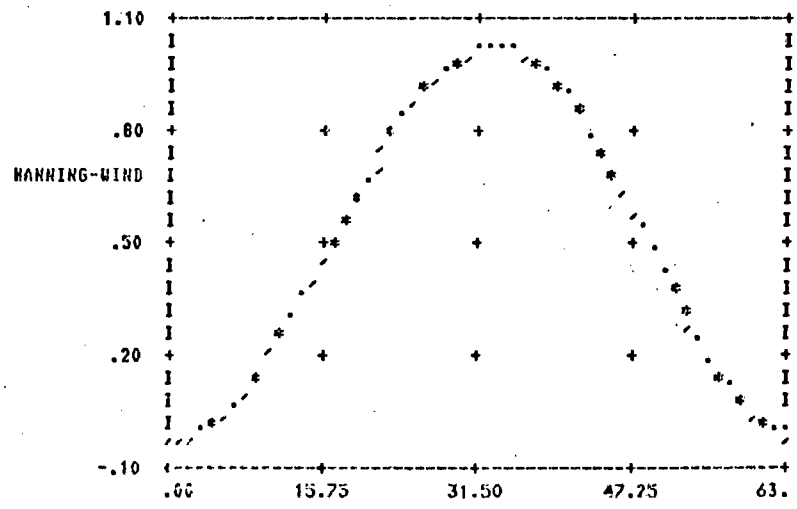
A3.5 SUBROUTINES

The subroutines called by WAVEPACK are listed. The detailed operation of each subroutine is fairly well described by the source code explanatory notes. For specific details the reader is referred to the source code listings.

A3.6 USING THE ANALYSIS PACKAGE

A typical sequence of commands required to initiate program execution is given overleaf. It assumes that all the files required have already been catalogued. Some simplification may be achieved by incorporating these commands into a file element for use with the @ADD processor. For further details of this, refer to the UNIVAC 1106 operating manual.

```
*DESTROY USERID/PASSWORD ENTRY*
*UNIVAC 1100 OPERATING SYSTEM  LEV. 36R1A/UCT85 (RSI)*
>@RUN,/N runid,A0510-R115,DOVEMD
DATE: 091581      TIME: 160431
>@ASG,A MIKED.
READY
>@ASG,A SAVE1.
READY
>@USE 11.,SAVE1.
READY
>@ASG,A SAVE2.
READY
>@USE 12.,SAVE2.
READY
>@ASG,A SAVE3.
READY
>@USE 13.,SAVE3.
READY
>@ASG,A SAVE4.
READY
>@USE 14.,SAVE4.
READY
>@ASG,A SAVE5.
READY
>@USE 15.,SAVE5.
READY
>@ASG,A PRINTFILE.
READY
>@USE 21.,PRINTFILE.
READY
>@GDP*ABS.INPUT PLOTFILE.
GDP/V2      09/15/81 16:08:44
>@XQT MIKED.WAVEPACK

  WAVEPACK   V1.3     12/6/81



  Type HELP to list commands
  ENTER COMMAND
>FILL,WA1,1.,0,63

  WA1 set equal to     1.000000 from   0 to  63 inclusive.
  ENTER COMMAND
>WINDOW,WA1

  Window codes are :
    1 - Zero order window.
    2 - First order window.
    3 - Hanning window.
    4 - Hamming window.
    5 - Blackman window.

  ENTER  window code,window centre  >>>
>3,32

  Window number 3 applied to WA1
  ENTER COMMAND
>SLABEL,WA1,HANNING-WIND
  ENTER COMMAND
>GRAPH,WA1
```



```
>PRINT,WA1
```

HANNING-WIND DATA

DATA INDEX :	0	1	2	3	4	5	6	7
DATA :	.000506	.000000	.002449	.009773	.021899	.038709	.060038	.085677
DATA INDEX :	8	9	10	11	12	13	14	15
DATA :	.115375	.148842	.185748	.225733	.268405	.313346	.360116	.408256
DATA INDEX :	16	17	18	19	20	21	22	23
DATA :	.457294	.506751	.556142	.604983	.652796	.699111	.743476	.785456
DATA INDEX :	24	25	26	27	28	29	30	31
DATA :	.824639	.860642	.893111	.921730	.946217	.966332	.981879	.992705
DATA INDEX :	32	33	34	35	36	37	38	39
DATA :	.998704	.999818	.996035	.987392	.973975	.955914	.933387	.906614
DATA INDEX :	40	41	42	43	44	45	46	47
DATA :	.875857	.841419	.803635	.762877	.719544	.674060	.626871	.578438
DATA INDEX :	48	49	50	51	52	53	54	55
DATA :	.529238	.479751	.430462	.381854	.334404	.288576	.244820	.203563
DATA INDEX :	56	57	58	59	60	61	62	63
DATA :	.165211	.130138	.098689	.071171	.047855	.028967	.014695	.005177

```
  ENTER COMMAND
>FILL,WA1,0.,0,63

  WA1 set equal to      .000000 from   0 to  63 inclusive.
  ENTER COMMAND
>FILL,WA1,1.,0,9

  WA1 set equal to     1.000000 from   0 to   9 inclusive.
  ENTER COMMAND
>FFT

  FFT computed with Re = WA1 and Im = WA2
  ENTER COMMAND
>PLOT,WA1

  GDP plotting interface.

  ENTER >>  xaxis-label
>FREQUENCY

  ENTER >>  yaxis-label
>MAGNITUDE

  ENTER >>  plot-title
>FT-PULSE

  ENTER >>  axes length in cms. xaxis-length,yaxis-length
>15.,10.

  ENTER >>  xmax,xmin,ymax,ymin
>63.,0.,12.,-2.
```

```
  ENTER >>  GDP page name for this plot
>FT-PULSE

  WA1 plotted.
  ENTER COMMAND
>FILL,WA3,1.414,0,21

  WA3 set equal to     1.414000 from   0 to  21 inclusive.
  ENTER COMMAND
>SUBST,WA3,19

  SUBSTITUTE MODE - type E to exit.

              ( 19) =    1.414000
>

              ( 20) =    1.414000
>34.9932

              ( 20) =   34.993200
>

              ( 21) =    1.414000
>0.

              ( 21) =     .000000
>

              ( 22) =     .000000
>E

  SUBSTITUTE MODE  terminated.
  ENTER COMMAND
>STOP

  WAVEPACK  TERMINATED     goodbye!


>@FIN
```

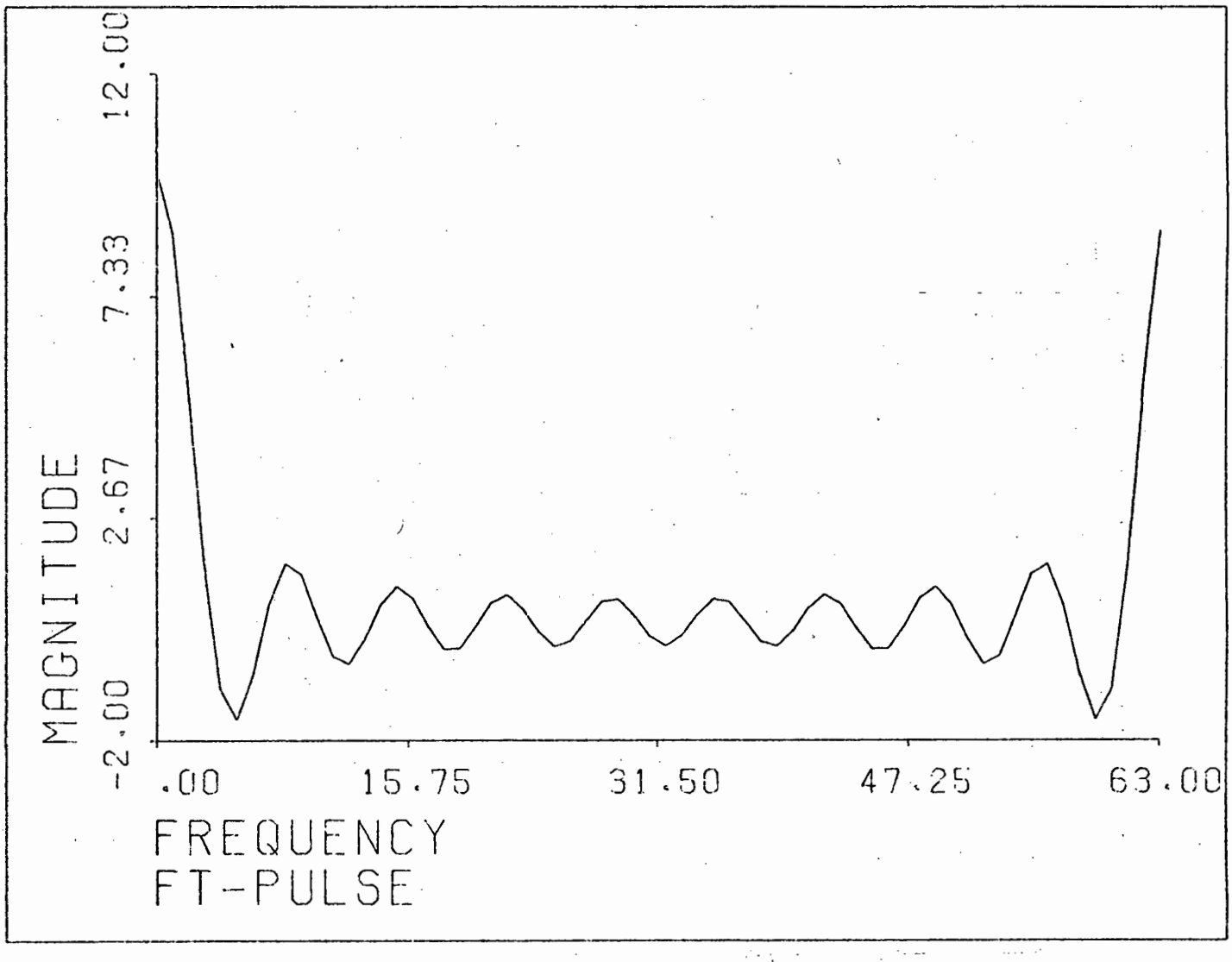


Figure A3.2 The graph generated by the runstream given here having been sent to the Calcomp plotter.

With reference to the example terminal session all the commands entered by the user are preceded by a prompt (>) from the computer. The other messages are generated by the program. The first command used is FILL which sets all the data elements in work area 1 equal to unity. The Hanning window is then applied to this data with the window centre at 32. The work area is then labelled and plotted using the character graph. The data in the work area is then listed. The following pair of FILL commands set all the work area data to zero and then a pulse shaped function is entered. The magnitude of the Fourier transform of this pulse function is then calculated and plotted. The graph generated using this runstream was plotted on the Calcomp plotter and is given in Figure A3.2. The various titles entered into the program are seen on the plotted curve. The axis length defines the size of the plot and for an A4 size plot should be approximately 25cm x 15cm. Having generated the required plot the Graphics Display Package is used to send the plot to the system plotter.

The use of the SUBST command is also illustrated. The various data windows and their Fourier transforms are given in Figure A3.4. An example of the convolution command is shown in Figure A3.5 where rectangular and triangular pulses are convolved.

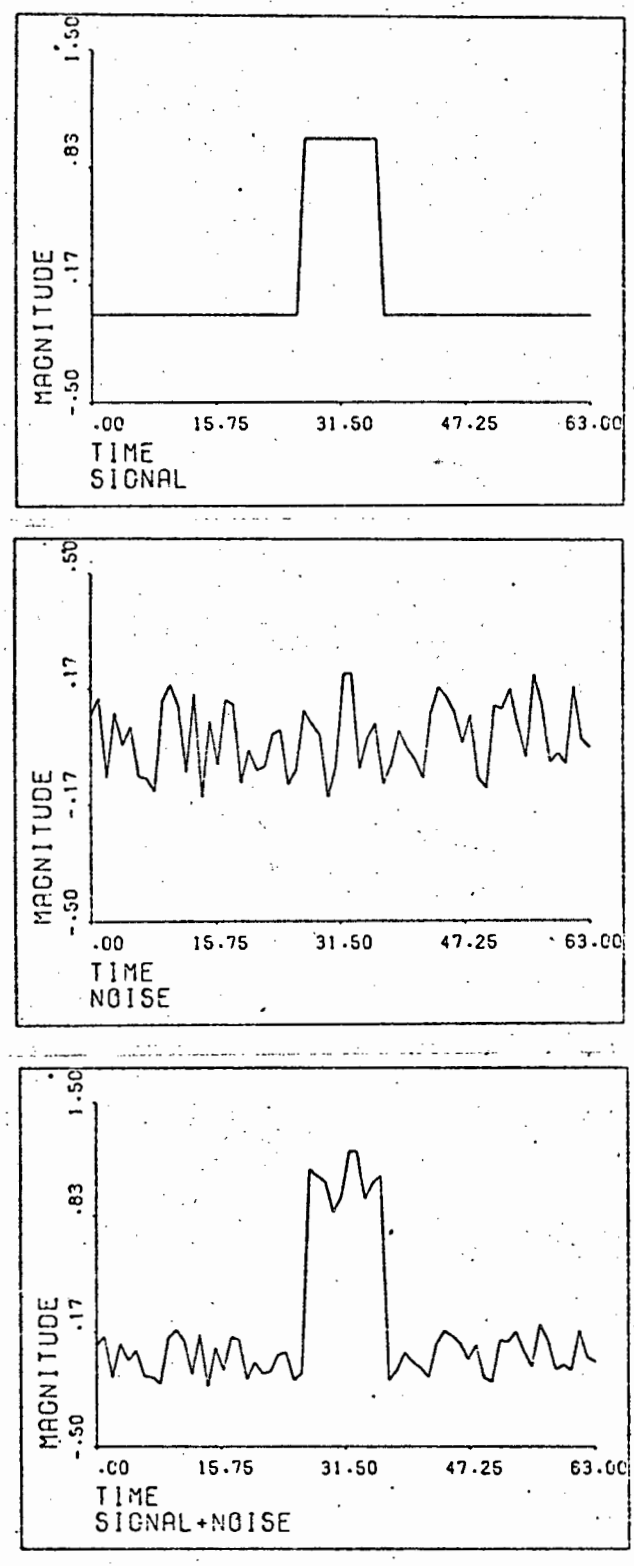


Figure A3.3 An illustration of the NOISE+ function.

A3.7 CONCLUSION

The development of this analysis package has generated some interest by other parties. The GRAPHV subroutine has been adopted by the University of Cape Town Computing Service as a library routine. Research into the application of digital filtering techniques to problems arising in radar is under way using WAVEPACK. It may thus be concluded that the facilities which are provided by the program are generally useful.

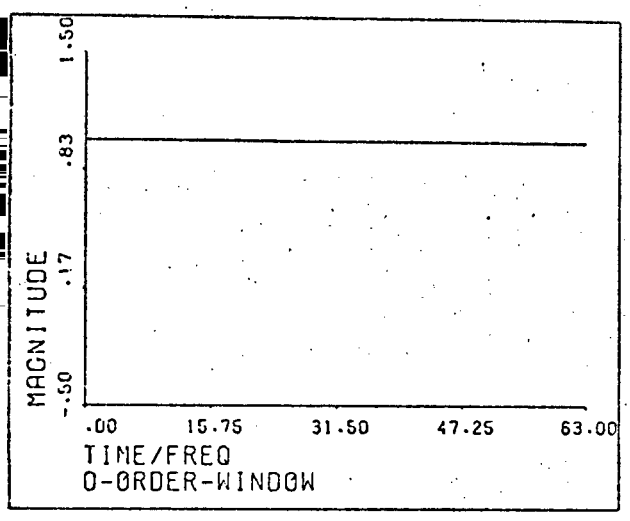


Figure A3.4a The zero-order window.

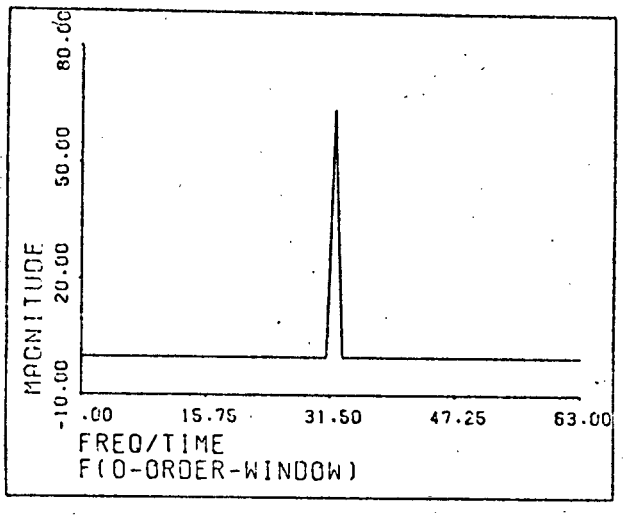


Figure A3.4b The Fourier transform of the zero-order window.

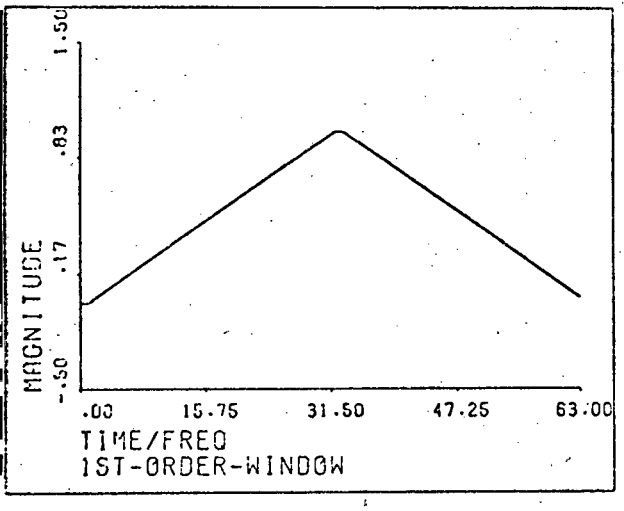


Figure A3.4c The first-order window.

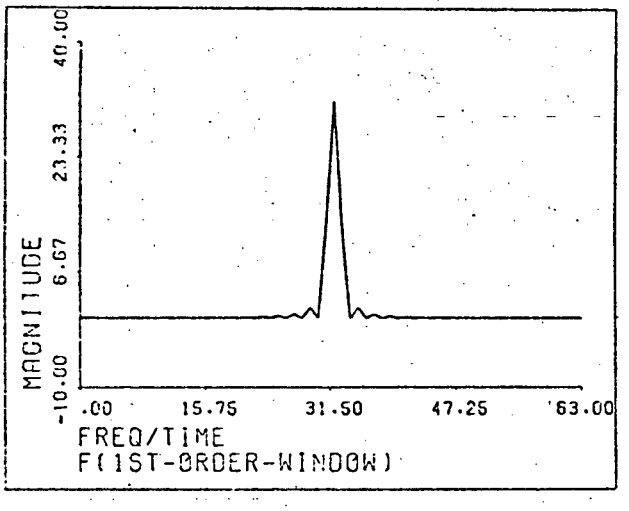


Figure A3.4d The Fourier transform of the first-order window.

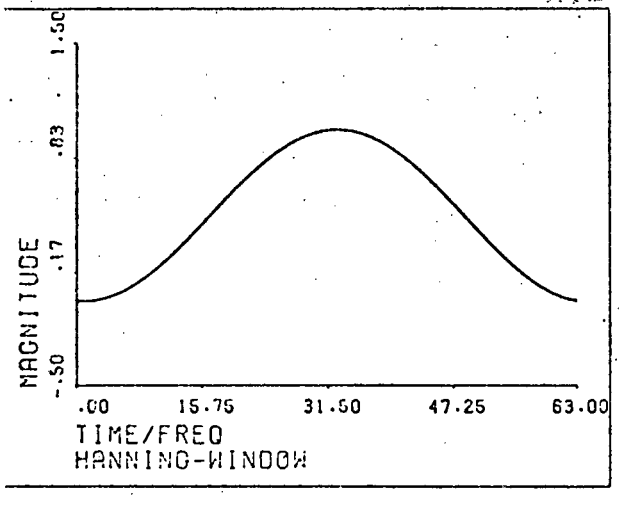


Figure A3.4e The Hanning window.

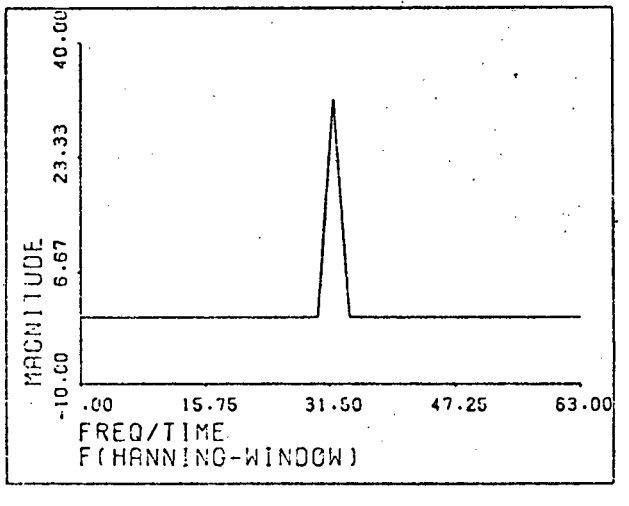


Figure A3.4f The Fourier transform of the Hanning window.

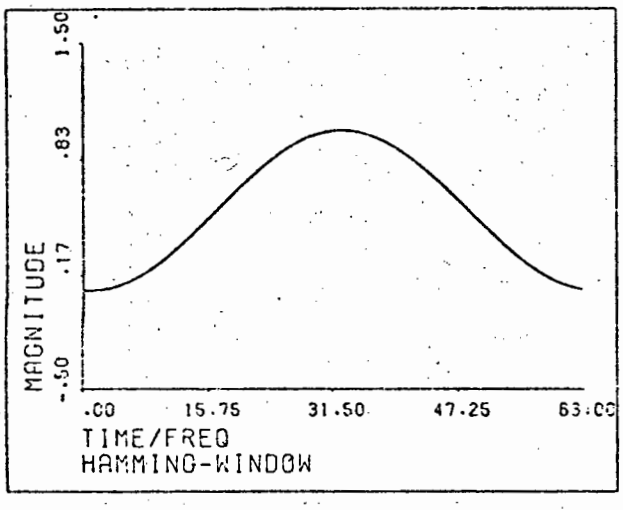


Figure A3.4g The Hamming window.

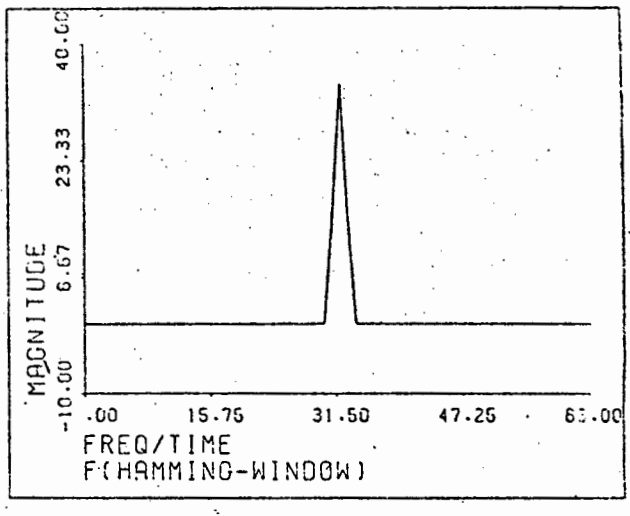


Figure A3.4h The Fourier transform of the Hamming window.

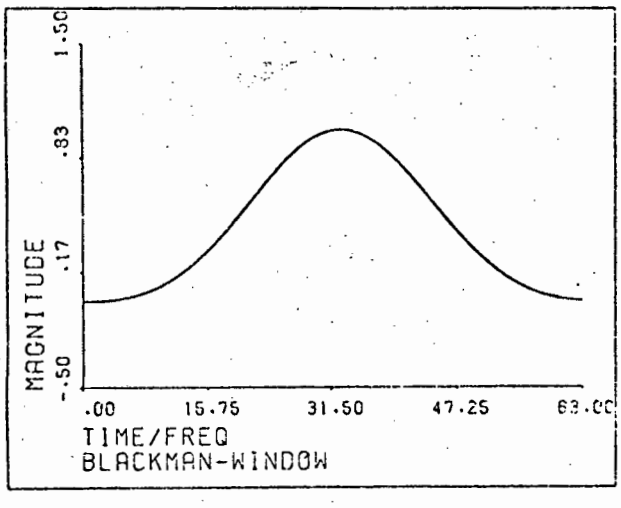


Figure A3.4i The Blackman window.

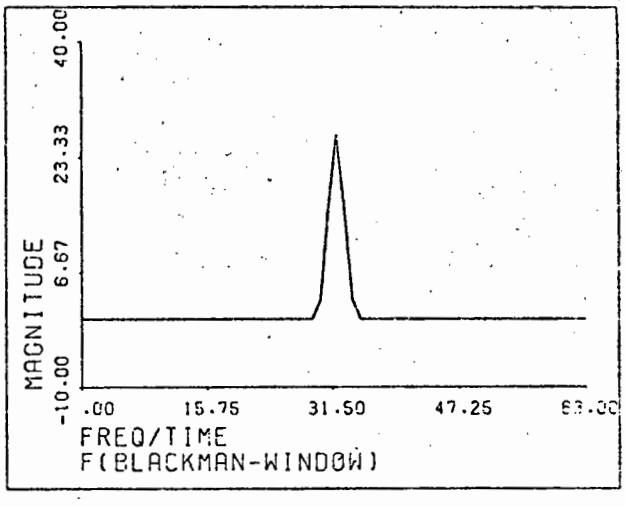


Figure A3.4j The Fourier transform of the Blackman window.

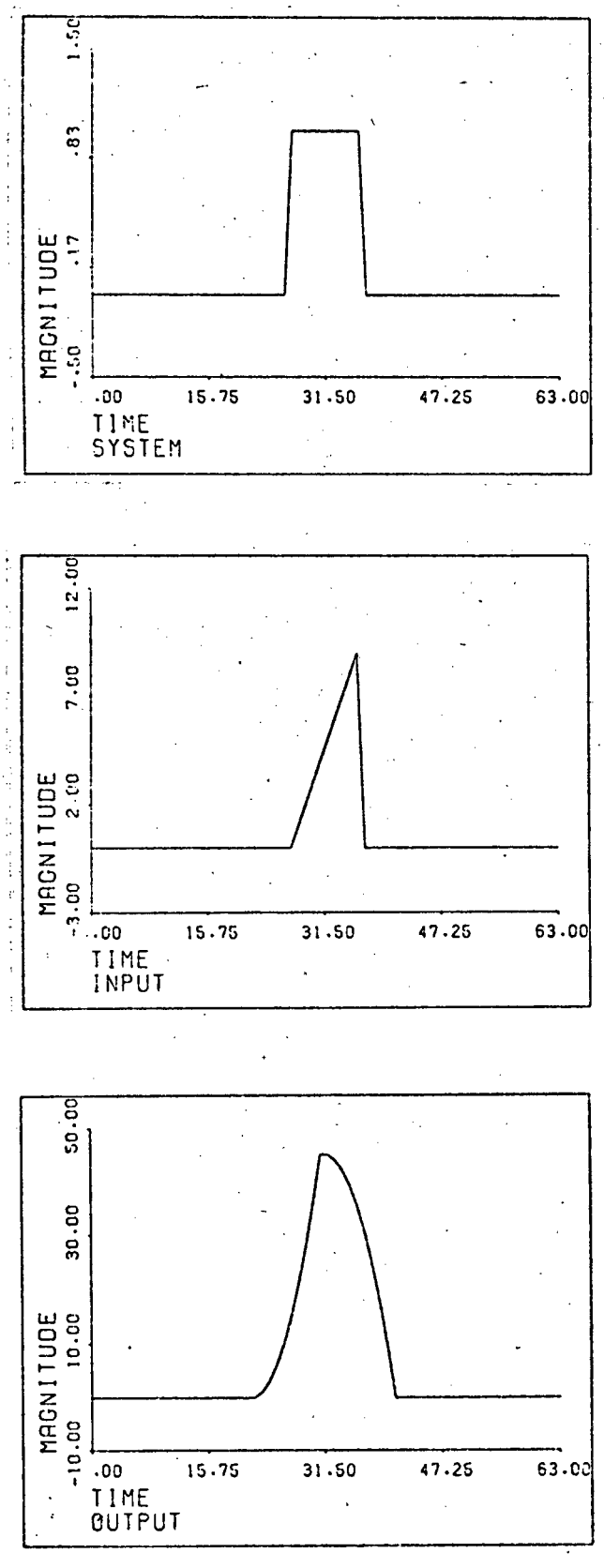


Figure A3.5 Convolution of rectangular and triangular pulses.

```
C
C
C  TITLE:  WAVEPACK
C
C
C
C  DESCRIPTION:
C
C  This program is designed to function as an interactive waveform
C  analysis package.  Contained in this package are facilities for
C  implementing digital signal processing.  Motivation for its in-
C  ception has been to  deconvolve  two  waveforms  using  Fourier
C  transform techniques.  For the program to operate four  Fortran
C  files are needed and one printfile and a plotfile.
C
C
C  CALLS:
C
C  FFT, INVFFT, LINFLD, GRAPHV, GRAPHW, PRINT, WINDOW, NEWPAG,
C  PAGNAM, SLICIT.
C
C  CODE:
C
C
C    Defining variables used.
C    ========================
C
       IMPLICIT  COMPLEX (A - Z)
C
       PARAMETER  NPTS = 64, M = 6
       PARAMETER  NFLDS = 5, CLNGTH = 80, FLNGTH = 20, NSTORS = 10
       PARAMETER  PFILE = 21, NWAREA = 4
C
       CHARACTER  TITLE*12 (0:NSTORS-1), TITWA*12 (1:NWAREA)
       CHARACTER  COMAND*80, FIELD*20 (1:NFLDS), XEXP*72
       CHARACTER * 20  XLABEL, YLABEL, PTITLE
       CHARACTER  GDPNAM*12
C
       INTEGER  A, B, I, J, ERRFLG, WAINDX, STORNO
       INTEGER  WNUM, CENTRE, SHIFT, N, NTIMES
       INTEGER  NXCHAR, NYCHAR, NTCHAR
C
       REAL  AXIS(0:NPTS-1), WAREA(1:NWAREA,0:NPTS-1)
       REAL  DATA(0:NPTS-1), STORE(0:NSTORS-1,0:NPTS-1)
       REAL  WDATA(1:NPTS), TEMP, Re, Im, r, O
       REAL  REDATA(1:NPTS), SRSA(0:NPTS-1), PRSA(0:NPTS-1)
       REAL  XREAL(1:NPTS), XIMAG(1:NPTS), YREAL(1:NPTS), SREAL(1:NPTS)
       REAL  SIMAG(1:NPTS), GREAL(1:NPTS), GIMAG(1:NPTS), PREAL(1:NPTS)
       REAL  PIMAG(1:NPTS)
       REAL  XAXIS, YAXIS, XMIN, XMAX, YMIN, YMAX
C
C
C
C    Setting up axis for graphing results.
C    =====================================
C
       DO  1000  I = 0,NPTS-1
          AXIS (I) = I
1000   CONTINUE
C
C
C
C
C    Initializing labels.
C    ====================
C
       DO  2000  I = 0,NSTORS-1
          TITLE(I) = '
2000   CONTINUE
       DO  2010  I = 1,NWAREA
          TITWA(I) = TITLE(1)
2010   CONTINUE
C
C
C
C
C
C     Printing a heading.
C     ===================
C
C
       PRINT 98003
       PRINT 98004
C
C
C    Soliciting user for command.
C    ============================
C
C
90001  PRINT 98005
       READ ( 0, 98006, ERR=90001 )  COMAND
C
       CALL  LINFLD ( COMAND, CLNGTH, FIELD, FLNGTH, NFLDS, ERRFLG )
C
```

```
C
C    Testing for a command syntax error.
C    ===================================
C
C
      IF ( ERRFLG .NE. 0 )  THEN
         PRINT 99003
         GO TO 90001
      ENDIF
C
C
C
C
C
C
C=============================================
C+++++++++ TESTING  FOR THE COMMANDS +++++++++
C=============================================
C
C
C
C
C
C
C
C    Implementing the SAVE and RETN commands.
C    ========================================
C
C
C
      IF ( FIELD(1) .EQ. 'SAVE' .OR. FIELD(1) .EQ. 'RETN' )  THEN
C
C
C       Validating the contents of field three and setting WAINDX.
C       ----------------------------------------------------------
C
C
C
         IF ( FIELD(3) .EQ. 'WA1' )  THEN
            WAINDX = 1
         ELSE IF ( FIELD(3) .EQ. 'WA2' )  THEN
            WAINDX = 2
         ELSE IF ( FIELD(3) .EQ. 'WA3' )  THEN
            WAINDX = 3
         ELSE IF ( FIELD(3) .EQ. 'WA4' )  THEN
            WAINDX = 4
         ELSE
            WAINDX = 0
            PRINT 99001
         ENDIF
C
C
C
C
C       Validating and setting Fortran file number before
C       reading or writing to or from disk.
C       ------------------------------------------------
C
C
C
         IF ( WAINDX .NE. 0 )  THEN
C
            DECODE ( 98001, FIELD(2) )  STORNO
            STORNO = STORNO + 10
C
C
            IF ( STORNO .GE. 11 .AND. STORNO .LE. 30 )  THEN
C
               IF ( FIELD(1) .EQ. 'SAVE' )  THEN
                  DO  10010  I = 0,NPTS-8,8
                     WRITE ( STORNO, 98002 ) ( WAREA (WAINDX,I+J),
     &                                                  J = 0,7 )
10010             CONTINUE
                  PRINT 98003, WAINDX, STORNO-10
C
               ELSE
C
                  DO  10020  I = 0,NPTS-8,8
                     READ ( STORNO, 98002 ) ( WAREA (WAINDX,I+J),
     &                                                  J = 0,7 )
10020             CONTINUE
                  PRINT 98009, WAINDX, STORNO-10
C
               ENDIF
C
               REWIND  STORNO
C
            ELSE
C
               PRINT 99002
C
            ENDIF
C
         ENDIF
C
```

```
C
C
C     Implementing LABEL commands.
C     ============================
C
C
      ELSE IF ( FIELD(1) .EQ. 'SLABEL' )  THEN
C
         IF ( FIELD(2) .EQ. 'WA1' )  THEN
            TITWA(1) = FIELD(3)
         ELSE IF ( FIELD(2) .EQ. 'WA2' )  THEN
            TITWA(2) = FIELD(3)
         ELSE IF ( FIELD(2) .EQ. 'WA3' )  THEN
            TITWA(3) = FIELD(3)
         ELSE IF ( FIELD(2) .EQ. 'WA4' )  THEN
            TITWA(4) = FIELD(3)
         ELSE
            DECODE ( 98001, FIELD(2), ERR=11000 )  STORNO
            ERRFLG = 1
            DO  11010  I = 0,NSTORS-1
               IF ( STORNO .EQ. I )  ERRFLG = 0
11010       CONTINUE
            IF ( ERRFLG .NE. 0 )  GO TO 11000
            TITLE(STORNO) = FIELD(3)
            GO TO 11020
C
11000       PRINT 99005
11020       CONTINUE
C
         ENDIF
C
C
C
C
      ELSE IF ( FIELD(1) .EQ. 'RLABEL' )  THEN
C
         IF ( FIELD(2) .EQ. 'WA1' )  THEN
            PRINT 98010, 1, TITWA(1)
         ELSE IF ( FIELD(2) .EQ. 'WA2' )  THEN
            PRINT 98010, 2, TITWA(2)
         ELSE IF ( FIELD(2) .EQ. 'WA3' )  THEN
            PRINT 98010, 3, TITWA(3)
         ELSE IF ( FIELD(2) .EQ. 'WA4' )  THEN
            PRINT 98010, 4, TITWA(4)
         ELSE
            DECODE ( 98001, FIELD(2), ERR=11030 )  STORNO
            ERRFLG = 1
            DO  11040  I = 0,NSTORS-1
               IF ( STORNO .EQ. I )  ERRFLG = 0
11040       CONTINUE
            IF ( ERRFLG .NE. 0 )  GO TO 11030
            PRINT 98012, STORNO, TITLE(STORNO)
            GO TO 11050
C
11030       PRINT 99005
11050       CONTINUE
C
         ENDIF
C
C
C
         ELSE IF ( FIELD(1) .EQ. 'LLABEL' )  THEN
            PRINT 98013
            DO  11070  I = 1,NWAREA
               PRINT 98010, I, TITWA(I)
11070       CONTINUE
            DO  11060  I = 0,NSTORS-1
               PRINT 98012, I, TITLE(I)
11060       CONTINUE
C
C
C
C
C   Implementing STO and RCL commands.
C  ==================================
C
C
      ELSE IF ( FIELD(1) .EQ. 'STO' .OR. FIELD(1) .EQ. 'RCL' )  THEN
C
C
C        Testing the validity of field 3.
C       ---------------------------------
C
C
         IF ( FIELD(3) .EQ. 'WA1' )  THEN
            WAINDX = 1
         ELSE IF ( FIELD(3) .EQ. 'WA2' )  THEN
            WAINDX = 2
         ELSE IF ( FIELD(3) .EQ. 'WA3' )  THEN
            WAINDX = 3
         ELSE IF ( FIELD(3) .EQ. 'WA4' )  THEN
            WAINDX = 4
         ELSE
            PRINT 99001
            WAINDX = 0
         ENDIF
```

```
C
C
C       Having detected an error in field 3 exit section.
C       -------------------------------------------------
C
        IF ( WAINDX .EQ. 0 )  GO TO 12010
C
C
C       Testing to see if field 3 is a label.
C       -------------------------------------
C
        STORNO = -1
C
        DO  12020  I = 0,NSTORS-1
           IF ( FIELD(2) .EQ. TITLE(I) )  STORNO = I
12020   CONTINUE
C
C
C       If field 3 corresponds to label - store work area.
C       --------------------------------------------------
C
        IF ( STORNO .GE. 0 )  THEN
C
           IF ( FIELD(1) .EQ. 'STO' )  THEN
              DO  12030  I = 0,NPTS-1
                 STORE(STORNO,I) = WAREA(WAINDX,I)
12030         CONTINUE
              PRINT 98014, WAINDX, TITLE(STORNO)
C
           ELSE
              DO  12040  I = 0,NPTS-1
                 WAREA(WAINDX,I) = STORE(STORNO,I)
12040         CONTINUE
              TITWA(WAINDX) = TITLE(STORNO)
              PRINT 98015, WAINDX, TITLE(STORNO)
C
           ENDIF
C
        ELSE
C
           DECODE ( 98001, FIELD(2), ERR=12050 )  STORNO
           IF ( STORNO .LT. 0 .OR. STORNO .GT. NSTORS-1 )  GO TO 12060
C
           IF ( FIELD(1) .EQ. 'STO' )  THEN
              DO  12070  I = 0,NPTS-1
                 STORE(STORNO,I) = WAREA(WAINDX,I)
12070         CONTINUE
              PRINT 98016, WAINDX, STORNO
C
           ELSE
C
              DO  12080  I = 0,NPTS-1
                 WAREA(WAINDX,I) = STORE(STORNO,I)
12080         CONTINUE
              TITWA(WAINDX) = TITLE(STORNO)
              PRINT 98017, WAINDX, STORNO
C
           ENDIF
C
           GO TO 12090
C
12050      PRINT 99005
           GO TO 12090
C
12060      PRINT 99002
12090      CONTINUE
C
        ENDIF
C
12010   CONTINUE
C
C
C
C
C
C   Implementing PRINT and GRAPH commands.
C   ======================================
C
C
     ELSE IF ( FIELD(1) .EQ. 'PRINT' .OR. FIELD(1) .EQ.
    &          'PRINTF' .OR. FIELD(1) .EQ. 'GRAPH'
    &          .OR. FIELD(1) .EQ. 'GRAPHF' )  THEN
C
C
        IF ( FIELD(2) .EQ. 'WA1' )  THEN
           WAINDX = 1
        ELSE IF ( FIELD(2) .EQ. 'WA2' )  THEN
           WAINDX = 2
        ELSE IF ( FIELD(2) .EQ. 'WA3' )  THEN
           WAINDX = 3
        ELSE IF ( FIELD(2) .EQ. 'WA4' )  THEN
           WAINDX = 4
        ELSE
           WAINDX = 0
           PRINT 99005
```

```
C
C
C       Having detected an error in field 2 exit section.
C       -------------------------------------------------
C
         IF ( WAINDX .NE. 0 )  THEN
C
C         Interfacing a 2 dimensional array with PRINT and GRAPHW.
C         --------------------------------------------------------
C
            DO  13020  I = 0,NPTS-1
               DATA(I) = WAREA(WAINDX,I)
13020       CONTINUE
C
            IF ( FIELD(1) .EQ. 'PRINT' )  THEN
               CALL  PRINT ( DATA, NPTS, TITWA(WAINDX), 5 )
C
            ELSE IF ( FIELD(1) .EQ. 'PRINTF' )  THEN
               CALL  PRINT ( DATA, NPTS, TITWA(WAINDX), PFILE )
C
            ELSE IF ( FIELD(1) .EQ. 'GRAPH' )  THEN
               CALL  GRAPHW ( AXIS, DATA, TITWA(WAINDX),
     &                        XEXP, NPTS, 5 )
C
            ELSE
               CALL  GRAPHW ( AXIS, DATA, TITWA(WAINDX),
     &                        XEXP, NPTS, PFILE )
C
            ENDIF
C
C
         ENDIF
C
C
C
C
C
C   Implementing FFT (Fast Fourier Transform) command.
C  ====================================================
C
C
      ELSE IF ( FIELD(1) .EQ. 'FFT' )  THEN
C
         DO  23010  I = 0,NPTS-1
            REDATA(I+1) = WAREA(1,I)
            WDATA(I+1) = WAREA(2,I)
23010    CONTINUE
C
         CALL  FFT ( REDATA, WDATA, M )
C
         DO  23020  I = 0,NPTS-1
            WAREA(1,I) = REDATA(I+1)
            WAREA(2,I) = WDATA(I+1)
23020    CONTINUE
C
         PRINT 98018
C
C
C
C   Implementing INVFFT (Inverse Fast Fourier Transform) command.
C  ================================================================
C
C
      ELSE IF ( FIELD(1) .EQ. 'INVFFT' )  THEN
C
         DO  23030  I = 0,NPTS-1
            REDATA(I+1) = WAREA(1,I)
            WDATA(I+1) = WAREA(2,I)
23030    CONTINUE
C
         CALL  INVFFT ( REDATA, WDATA, M )
C
         DO  23040  I = 0,NPTS-1
            WAREA(1,I) = REDATA(I+1)
            WAREA(2,I) = WDATA(I+1)
23040    CONTINUE
C
         PRINT 98019
C
C
```

```
C
C   Implementing WINDOW command.
C   ============================
C
C
      ELSE IF ( FIELD(1) .EQ. 'WINDOW' )  THEN
C
         IF ( FIELD(2) .EQ. 'WA1' )  THEN
            WAINDX = 1
         ELSE IF ( FIELD(2) .EQ. 'WA2' )  THEN
            WAINDX = 2
         ELSE IF ( FIELD(2) .EQ. 'WA3' )  THEN
            WAINDX = 3
         ELSE IF ( FIELD(2) .EQ. 'WA4' )  THEN
            WAINDX = 4
         ELSE
            PRINT 99005
            WAINDX = 0
         ENDIF
C
C
C        Having detected an error in field 2 exit section.
C        -------------------------------------------------
C
         IF ( WAINDX .EQ. 0 )  GO TO 14010
C
C
C        Printing window menu.
C        ---------------------
C
         PRINT 98020
C
         READ (8, 98006)  COMAND
         CALL  LINFLD ( COMAND, CLNGTH, FIELD, FLNGTH, NFLDS, ERRFLG )
C
         DECODE ( 98001, FIELD(1), ERR=14020 ) WNUM
         DECODE ( 98001, FIELD(2), ERR=14030 ) CENTRE
C
C
C       Adjusting data to be in correct position before windowing.
C       ----------------------------------------------------------
C
         IF ( CENTRE .GE. 0 .AND. CENTRE .LT. NPTS/2-1 )  THEN
C
            DO  14040  I = 1, NPTS/2-1-CENTRE
               TEMP = WAREA(WAINDX,NPTS-1)
               DO  14050  J = 0,NPTS-2
                  WAREA(WAINDX,NPTS-1-J) = WAREA(WAINDX,NPTS-2-J)
14050          CONTINUE
               WAREA(WAINDX,0) = TEMP
14040       CONTINUE
C
         ELSE IF ( CENTRE .GT. NPTS/2-1 .AND. CENTRE .LE. NPTS-1 )  THEN
C
            DO  14060  I = 1,CENTRE+1-NPTS/2
               TEMP = WAREA(WAINDX,0)
               DO  14070  J = 0,NPTS-2
                  WAREA(WAINDX,J) = WAREA(WAINDX,J+1)
14070          CONTINUE
            WAREA(WAINDX,NPTS-1) = TEMP
14060       CONTINUE
C
         ELSE
            CONTINUE
         ENDIF
C
C
C
C
C
C
C       Entering data to be windowed into an array with slightly
C       different dimensions to interface with the WINDOW subroutine.
C       -------------------------------------------------------------
C
         DO  14130  I = 0,NPTS-1
            WDATA(I+1) = WAREA(WAINDX,I)
14130    CONTINUE
C
         CALL  WINDOW ( WDATA, WDATA, NPTS, NPTS, WNUM )
C
C
C       Replacing windowed data into work area.
C       ---------------------------------------
C
         DO  14080  I = 0,NPTS-1
            WAREA(WAINDX,I) = WDATA(I+1)
14080    CONTINUE
C
```

```
C
C       Adjusting data back to original position.
C       ----------------------------------------
C
        IF ( CENTRE .GE. 0 .AND. CENTRE .LT. NPTS/2-1 )  THEN
C
           DO  14090  I = 1,NPTS/2-1-CENTRE
              TEMP = WAREA(WAINDX,0)
              DO  14100  J = 0,NPTS-2
                 WAREA(WAINDX,J) = WAREA(WAINDX,J+1)
14100         CONTINUE
              WAREA(WAINDX,NPTS-1) = TEMP
14090      CONTINUE
C
        ELSE IF ( CENTRE .GT. NPTS/2-1 .AND. CENTRE .LE. NPTS-1 )  THEN
C
           DO  14110  I = 1,CENTRE+1-NPTS/2
              TEMP = WAREA(WAINDX,NPTS-1)
              DO  14120  J = 0,NPTS-2
                 WAREA(WAINDX,NPTS-1-J) = WAREA(WAINDX,NPTS-2-J)
14120         CONTINUE
           WAREA(WAINDX,0) = TEMP
14110      CONTINUE
C
        ELSE
           CONTINUE
C
        ENDIF
C
C
C
        PRINT 98021, WNUM, WAINDX
C
        GO TO 14010
C
14020   PRINT 99006
        GO TO 14010
C
14030   PRINT 99005
14010   CONTINUE
C
C
C
C
C
C
C
C
C   Implementing R-P and P-R commands.
C   =================================
C
     ELSE IF ( FIELD(1) .EQ. 'R-P' )  THEN
C
        DO  15010  I = 0,NPTS-1
           r = SQRT( WAREA(1,I)**2 + WAREA(2,I)**2 )
           0 = ATAN( WAREA(2,I)/WAREA(1,I) )
           IF ( WAREA(1,I) .LT. 0 .AND. WAREA(2,I) .GT. 0 )
     &        0 = 0 + 3.141592654
           IF ( WAREA(1,I) .LT. 0 .AND. WAREA(2,I) .LT. 0 )
     &        0 = 0 - 3.141592654
           WAREA(1,I) = r
           WAREA(2,I) = 0
15010   CONTINUE
        PRINT 98022
C
     ELSE IF ( FIELD(1) .EQ. 'P-R' )  THEN
C
        DO  15020  I = 0,NPTS-1
           Re = WAREA(1,I) * COS( WAREA(2,I) )
           Im = WAREA(1,I) * SIN( WAREA(2,I) )
           WAREA(1,I) = Re
           WAREA(2,I) = Im
15020   CONTINUE
        PRINT 98023
C
C
C
C
C
C   Implementing CONV command.
C   =========================
C
     ELSE IF ( FIELD(1) .EQ. 'CONV' )  THEN
C
        DO  17010  I = 0,NPTS-1
        DATA(I) = 0.
           DO  17020  J = 0,NPTS-1
              SHIFT = I-J
              IF ( SHIFT .LT. 0 )  SHIFT = SHIFT + NPTS
              DATA(I) = DATA(I) + WAREA(1,J) * WAREA(2,SHIFT)
17020      CONTINUE
17010   CONTINUE
C
C
```

```
C       Setting WA3 equal to the resulting convolution.
C       -----------------------------------------------
C
         DO  17030  I = 0,NPTS-1
            WAREA(3,I) = DATA(I)
17030    CONTINUE
         PRINT 98024
C
C
C
C
C
C    Implementing FOLD command.
C    ==========================
C
       ELSE IF ( FIELD(1) .EQ. 'FOLD' )  THEN
C
         IF ( FIELD(2) .EQ. 'WA1' )  THEN
            WAINDX = 1
         ELSE IF ( FIELD(2) .EQ. 'WA2' )  THEN
            WAINDX = 2
         ELSE IF ( FIELD(2) .EQ. 'WA3' )  THEN
            WAINDX = 3
         ELSE IF ( FIELD(2) .EQ. 'WA4' )  THEN
            WAINDX = 4
         ELSE
            PRINT 99005
            WAINDX = 0
         ENDIF
C
         IF ( WAINDX .NE. 0 )  THEN
            DO  16010  I = 1,NPTS/2-1
               WAREA(WAINDX,NPTS-1-I) = WAREA(WAINDX,I)
16010       CONTINUE
            PRINT 98025, WAINDX
         ENDIF
C
C
C
C
C
C    Implementing FILL command.
C    ==========================
C
       ELSE IF ( FIELD(1) .EQ. 'FILL' ) THEN
C
         IF ( FIELD(2) .EQ. 'WA1' )  THEN
            WAINDX = 1
         ELSE IF ( FIELD(2) .EQ. 'WA2' )  THEN
            WAINDX = 2
         ELSE IF ( FIELD(2) .EQ. 'WA3' )  THEN
            WAINDX = 3
         ELSE IF ( FIELD(2) .EQ. 'WA4' )  THEN
            WAINDX = 4
         ELSE
            PRINT 99005
            WAINDX = 0
         ENDIF
C
         IF ( WAINDX .NE. 0 )  THEN
            DECODE ( 98001, FIELD(4), ERR=18010 ) A
            DECODE ( 98001, FIELD(5), ERR=18020 ) B
            DECODE ( 98001, FIELD(3), ERR=18030 ) TEMP
C
            DO  18040  I = A,B
               WAREA(WAINDX,I) = TEMP
18040       CONTINUE
            PRINT 98026, WAINDX, TEMP, A, B
            GO TO 18050
C
18010       PRINT 99007
            GO TO 18050
C
18020       PRINT 99008
            GO TO 18050
C
18030       PRINT 99001
18050       CONTINUE
         ENDIF
C
C
C
C
C
C    Implementing the EXCH command.
C    ==============================
C
       ELSE IF ( FIELD(1) .EQ. 'EXCH' )  THEN
C
         DO  19010  I = 0,NPTS-1
            TEMP = WAREA(1,I)
            WAREA(1,I) = WAREA(2,I)
            WAREA(2,I) = TEMP
19010    CONTINUE
         PRINT 98027
C
```

```
C
C
C
C
C    Implementing NORM command.
C    ==========================
C
      ELSE IF ( FIELD(1) .EQ. 'NORM' )  THEN
C
         IF ( FIELD(2) .EQ. 'WA1' )  THEN
            WAINDX = 1
         ELSE IF ( FIELD(2) .EQ. 'WA2' )  THEN
            WAINDX = 2
         ELSE IF ( FIELD(2) .EQ. 'WA3' )  THEN
            WAINDX = 3
         ELSE IF ( FIELD(2) .EQ. 'WA4' )  THEN
            WAINDX = 4
         ELSE
            PRINT 99005
            WAINDX = 0
         ENDIF
C
         IF ( WAINDX .NE. 0 )  THEN
            TEMP = ABS( WAREA(WAINDX,0) )
            DO  20010  I = 1,NPTS-1
               IF ( ABS( WAREA(WAINDX,I) ) .GT. TEMP )
     &            TEMP = ABS( WAREA(WAINDX,I) )
20010       CONTINUE
C
            DO  20020  I = 0,NPTS-1
               WAREA(WAINDX,I) = WAREA(WAINDX,I) / TEMP
20020       CONTINUE
C
            PRINT 98029, WAINDX, TEMP
C
         ENDIF
C
C
C
C
C
C    Implementing the SINE and ARCSINE commands.
C    ===========================================
C
      ELSE IF ( FIELD(1) .EQ. 'SINE' .OR. FIELD(1) .EQ. 'ARCSINE' ) THEN
C
         IF ( FIELD(2) .EQ. 'WA1' )  THEN
            WAINDX = 1
         ELSE IF ( FIELD(2) .EQ. 'WA2' )  THEN
            WAINDX = 2
         ELSE IF ( FIELD(2) .EQ. 'WA3' )  THEN
            WAINDX = 3
         ELSE IF ( FIELD(2) .EQ. 'WA4' )  THEN
            WAINDX = 4
         ELSE
            PRINT 99005
            WAINDX = 0
         ENDIF
C
         IF ( WAINDX .NE. 0 )  THEN
C
            TEMP = ABS( WAREA(WAINDX,0) )
            DO  21010  I = 1,NPTS-1
               IF ( ABS( WAREA(WAINDX,I) ) .GT. TEMP )
     &                        TEMP = ABS( WAREA(WAINDX,I) )
21010       CONTINUE
C
            IF ( FIELD(1) .EQ. 'SINE' )  THEN
               DO  21020  I = 0,NPTS-1
                  WAREA(WAINDX,I) = TEMP * SIN( (WAREA(WAINDX,I)/TEMP)
     &                                    * 1.570796327 )
21020          CONTINUE
               PRINT 98030, WAINDX
C
            ELSE
               DO  21030  I = 0,NPTS-1
                  WAREA(WAINDX,I) = TEMP * 0.636619772
     &                    * ASIN( WAREA(WAINDX,I) / TEMP )
21030          CONTINUE
               PRINT 98031, WAINDX
C
            ENDIF
C
         ENDIF
C
C
C
C
```

```
C
C    Implementing the ROTATE command.
C    ================================
C
      ELSE IF ( FIELD(1) .EQ. 'ROTATE' )  THEN
C
         IF ( FIELD(2) .EQ. 'WA1' )  THEN
            WAINDX = 1
         ELSE IF ( FIELD(2) .EQ. 'WA2' )  THEN
            WAINDX = 2
         ELSE IF ( FIELD(2) .EQ. 'WA3' )  THEN
            WAINDX = 3
         ELSE IF ( FIELD(2) .EQ. 'WA4' )  THEN
            WAINDX = 4
         ELSE
            PRINT 99005
            WAINDX = 0
         ENDIF
C
         IF ( WAINDX .NE. 0 )  THEN
C
            DECODE ( 98001, FIELD(3), ERR=22010 ) A
C
            IF ( A .GT. 0 )  THEN
               DO  22020  I = 1,A
               TEMP = WAREA(WAINDX,NPTS-1)
                  DO  22030  J = 0,NPTS-2
                     WAREA(WAINDX,NPTS-1-J) = WAREA(WAINDX,NPTS-2-J)
22030             CONTINUE
                  WAREA(WAINDX,0) = TEMP
22020          CONTINUE
               PRINT 98032, WAINDX, A
C
            ELSE IF ( A .LT. 0 )  THEN
               DO  22040  I = 1,ABS(A)
                  TEMP = WAREA(WAINDX,0)
                  DO  22050  J = 0,NPTS-2
                     WAREA(WAINDX,J) = WAREA(WAINDX,J+1)
22050             CONTINUE
                  WAREA(WAINDX,NPTS-1) = TEMP
22040          CONTINUE
               PRINT 98033, WAINDX, ABS(A)
C
            ELSE
               CONTINUE
            ENDIF
C
C
            GO TO 22060
C
22010       PRINT 99001
22060       CONTINUE
C
         ENDIF
C
C
C
C
C
C    Implementing arithmetic commands.
C    =================================
C
C
C
C    Implementing WA1 = WA2 and WA2 = WA1.
C    -------------------------------------
C
      ELSE IF ( FIELD(1) .EQ. 'WA1=WA2' )  THEN
C
         DO  25010  I = 0,NPTS-1
            WAREA(1,I) = WAREA(2,I)
25010    CONTINUE
         PRINT 98035, 1, 2
C
C
      ELSE IF ( FIELD(1) .EQ. 'WA2=WA1' )  THEN
C
         DO  25020  I = 0,NPTS-1
            WAREA(2,I) = WAREA(1,I)
25020    CONTINUE
         PRINT 98035, 2, 1
C
C
C
C    Implementing WA1 = WA3 and WA1 = WA4.
C    -------------------------------------
C
      ELSE IF ( FIELD(1) .EQ. 'WA1=WA3' )  THEN
C
         DO  25500  I = 0,NPTS-1
            WAREA(1,I) = WAREA(3,I)
25500    CONTINUE
         PRINT 98035, 1, 3
C
```

```
C
      ELSE IF ( FIELD(1) .EQ. 'WA1=WA4' )  THEN
C
         DO  25510  I = 0,NPTS-1
            WAREA(1,I) = WAREA(4,I)
25510    CONTINUE
         PRINT 98035, 1, 4
C
C
C
C    Implementing WA2=WA3 and WA2=WA4.
C    ---------------------------------
C
      ELSE IF ( FIELD(1) .EQ. 'WA2=WA3' )  THEN
C
         DO  25580  I = 0,NPTS-1
            WAREA(2,I) = WAREA(3,I)
25580    CONTINUE
         PRINT 98035, 2, 3
C
C
C
      ELSE IF ( FIELD(1) .EQ. 'WA2=WA4' )  THEN
C
         DO  25590  I = 0,NPTS-1
            WAREA(2,I) = WAREA(4,I)
25590    CONTINUE
         PRINT 98035, 2, 4
C
C
C
C
C    Implementing WA3 = WA1 and WA3 = WA2.
C    -------------------------------------
C
      ELSE IF ( FIELD(1) .EQ. 'WA3=WA1' )  THEN
C
         DO  25520  I = 0,NPTS-1
            WAREA(3,I) = WAREA(1,I)
25520    CONTINUE
         PRINT 98035, 3, 1
C
C
      ELSE IF ( FIELD(1) .EQ. 'WA3=WA2' )  THEN
C
         DO  25530  I = 0,NPTS-1
            WAREA(3,I) = WAREA(2,I)
25530    CONTINUE
         PRINT 98035, 3, 2
C
C
C
C    Implementing WA3 = WA4 and WA4 = WA1.
C    -------------------------------------
C
      ELSE IF ( FIELD(1) .EQ. 'WA3=WA4' )  THEN
C
         DO  25540  I = 0,NPTS-1
            WAREA(3,I) = WAREA(4,I)
25540    CONTINUE
         PRINT 98035, 3, 4
C
C
      ELSE IF ( FIELD(1) .EQ. 'WA4=WA1' )  THEN
C
         DO  25550  I = 0,NPTS-1
            WAREA(4,I) = WAREA(1,I)
25550    CONTINUE
         PRINT 98035, 4, 1
C
C
C
C
C    Implementing WA4 = WA2 and WA4 = WA3.
C    -------------------------------------
C
      ELSE IF ( FIELD(1) .EQ. 'WA4=WA2' )  THEN
C
         DO  25560  I = 0,NPTS-1
            WAREA(4,I) = WAREA(2,I)
25560    CONTINUE
         PRINT 98035, 4, 2
C
C
      ELSE IF ( FIELD(1) .EQ. 'WA4=WA3' )  THEN
C
         DO  25570  I = 0,NPTS-1
            WAREA(4,I) = WAREA(3,I)
25570    CONTINUE
         PRINT 98035, 4, 3
C
C
C
```

```
C
C
C    Setting selected areas of work area equal.
C    ------------------------------------------
C
      ELSE IF ( FIELD(1) .EQ. 'SWA1=WA2' .OR.
     &          FIELD(1) .EQ. 'SWA2=WA1' )  THEN
C
          DECODE ( 98001, FIELD(2), ERR=25030 ) A
          DECODE ( 98001, FIELD(3), ERR=25040 ) B
C
          IF ( A .GE. 0 .AND. A .LE. NPTS-1 .AND.
     &            B .GE. 0 .AND. B .LE. NPTS-1 )  THEN
C
             IF ( FIELD(1) .EQ. 'SWA1=WA2' )  THEN
                DO  25050  I = A,B
                   WAREA(1,I) = WAREA(2,I)
25050           CONTINUE
                PRINT 98036, A, B
C
             ELSE
                DO  25060  I = A, B
                   WAREA(2,I) = WAREA(1,I)
25060           CONTINUE
                PRINT 98037, A, B
             ENDIF
C
          ELSE
             PRINT 99001
             PRINT 99007
          ENDIF
C
          GO TO 25070
C
25030     PRINT 99005
          GO TO 25070
C
25040     PRINT 99001
25070     CONTINUE
C
C
C
C
C
C    Implementing  WA1 = WA1 x WA2
C    -----------------------------
C
      ELSE IF ( FIELD(1) .EQ. 'WA1=WA1*WA2' .OR. FIELD(1) .EQ.
     &          'WA1=WA2*WA1' )  THEN
C
          DO  25080  I = 0,NPTS-1
             WAREA(1,I) = WAREA(1,I) * WAREA(2,I)
25080     CONTINUE
          PRINT 98038
C
C
C
C
C
C    Implementing  WA1=WA1/WA2 and WA1=WA2/WA1.
C    ------------------------------------------
C
      ELSE IF ( FIELD(1) .EQ. 'WA1=WA1/WA2' )  THEN
C
          DO  25090  I = 0,NPTS-1
             WAREA(1,I) = WAREA(1,I)/WAREA(2,I)
25090     CONTINUE
          PRINT 98039
C
C
      ELSE IF ( FIELD(1) .EQ. 'WA1=WA2/WA1' )  THEN
C
          DO  25100  I = 0,NPTS-1
             WAREA(1,I) = WAREA(2,I)/WAREA(1,I)
25100     CONTINUE
          PRINT 98040
C
C
C
C
C
C    Implementing  WA1=WA1+WA2.
C    --------------------------
C
      ELSE IF ( FIELD(1) .EQ. 'WA1=WA1+WA2' .OR. FIELD(1)
     &          .EQ. 'WA1=WA2+WA1' )  THEN
C
          DO  25110  I = 0,NPTS-1
             WAREA(1,I) = WAREA(1,I) + WAREA(2,I)
25110     CONTINUE
          PRINT 98041
C
C
C
C
```

```
C
C    Implementing  WA1=WA1-WA2 and  WA1=WA2-WA1.
C    ------------------------------------------
C
      ELSE IF ( FIELD(1) .EQ. 'WA1=WA1-WA2' )  THEN
C
         DO  25120  I = 0,NPTS-1
            WAREA(1,I) = WAREA(1,I) - WAREA(2,I)
25120    CONTINUE
         PRINT 98042
C
C
      ELSE IF ( FIELD(1) .EQ. 'WA1=WA2-WA1' )  THEN
C
         DO  25130  I = 0,NPTS-1
            WAREA(1,I) = WAREA(2,I) - WAREA(1,I)
25130    CONTINUE
         PRINT 98043
C
C
C
C
C
C    Implementing  WA1=WA1*,constant.
C    --------------------------------
C
      ELSE IF ( FIELD(1) .EQ. 'WA1=WA1*' )  THEN
C
         DECODE ( 98001, FIELD(2), ERR=25300 ) TEMP
         DO  25140  I = 0,NPTS-1
            WAREA(1,I) = WAREA(1,I) * TEMP
25140    CONTINUE
         PRINT 98044, TEMP
C
         GO TO 25150
C
25300    PRINT 99005
25150    CONTINUE
C
C
C
C
C
C
C    Implementing  WA1=WA1/,constant.
C    --------------------------------
C
      ELSE IF ( FIELD(1) .EQ. 'WA1=WA1/' )  THEN
C
         DECODE ( 98001, FIELD(2), ERR=25160 ) TEMP
         DO  25170  I = 0,NPTS-1
            WAREA(1,I) = WAREA(1,I) / TEMP
25170    CONTINUE
         PRINT 98045,TEMP
C
         GO TO 25180
C
25160    PRINT 99005
25180    CONTINUE
C
C
C
C
C
C    Implementing  WA1=WA1+,constant.
C    --------------------------------
C
      ELSE IF ( FIELD(1) .EQ. 'WA1=WA1+' )  THEN
C
         DECODE ( 98001, FIELD(2), ERR=25190 ) TEMP
         DO  25200  I = 0,NPTS-1
            WAREA(1,I) = WAREA(1,I) + TEMP
25200    CONTINUE
         PRINT 98046, TEMP
C
         GO TO 25210
C
25190    PRINT 99005
25210    CONTINUE
C
C
C
C
```

```
C
C    Implementing  WA1=WA1-,constant.
C    --------------------------------
C
      ELSE IF ( FIELD(1) .EQ. 'WA1=WA1-' )  THEN
C
         DECODE ( 98001, FIELD(2), ERR=25220 ) TEMP
         DO  25230  I = 0,NPTS-1
            WAREA(1,I) = WAREA(1,I) - TEMP
25230    CONTINUE
         PRINT 98047, TEMP
C
      GO TO 25240
C
25220    PRINT 99005
25240    CONTINUE
C
C
C
C
C
C    Implementing  WA1=1/WA1.
C    ------------------------
C
      ELSE IF ( FIELD(1) .EQ. 'WA1=1/WA1' )  THEN
C
         DO  25250  I = 0,NPTS-1
            WAREA(1,I) = 1 / WAREA(1,I)
25250    CONTINUE
         PRINT 98048
C
C
C
C
C
C    Implementing SUBST command.
C    ===========================
C
      ELSE IF ( FIELD(1) .EQ. 'SUBST' )  THEN
C
         IF ( FIELD(2) .EQ. 'WA1' )  THEN
            WAINDX = 1
         ELSE IF ( FIELD(2) .EQ. 'WA2' )  THEN
            WAINDX = 2
         ELSE IF ( FIELD(2) .EQ. 'WA3' )  THEN
            WAINDX = 3
         ELSE IF ( FIELD(2) .EQ. 'WA4' )  THEN
            WAINDX = 4
         ELSE
            PRINT 99005
            WAINDX = 0
         ENDIF
C
C
         IF ( WAINDX .NE. 0 )  THEN
C
            DECODE ( 98001, FIELD(3), ERR=26050 ) A
            IF ( A .GE. 0 .AND. A .LE. NPTS-1 )  THEN
               PRINT 98049
               I = A
26040          PRINT 98050, TITWA(WAINDX), I, WAREA(WAINDX,I)
               READ ( 8, 98006) COMAND
               CALL  LINFLD ( COMAND, CLNGTH, FIELD, FLNGTH, NFLDS,
     &                                ERRFLG )
C
               IF ( FIELD(1) .EQ. 'E' )  GO TO 26010
C
               IF ( FIELD(1) .EQ. '  ' )  THEN
                  I = I +1
                  IF ( I .GT. NPTS-1 )  I = 0
               ELSE
                  DECODE ( 98001, FIELD(1), ERR=26020 ) TEMP
                  WAREA(WAINDX,I) = TEMP
                  GO TO 26030
C
26020             PRINT 99006
26030             CONTINUE
               ENDIF
C
               GO TO 26040
C
            ELSE
               PRINT 99009
            ENDIF
C
26010       PRINT 98051
            GO TO 26060
C
26050       PRINT 99001
26060       CONTINUE
C
         ENDIF
C
C
C
```

```
C
C
C    Implementing DECONVOLUTION commands.
C    ************************************
C
C
C      Implementing DECONV1.
C     ---------------------
C
      ELSE IF ( FIELD(1) .EQ. 'DECONV1' )  THEN
C
         DO  27010  I = 0,NPTS-1
            REDATA(I+1) = WAREA(1,I)
            WDATA(I+1) = WAREA(2,I)
27010    CONTINUE
C
         CALL  FFT ( REDATA, WDATA, M )
C
         DO  27020  I = 0,NPTS-1
            WAREA(1,I) = REDATA(I+1)
            WAREA(2,I) = WDATA(I+1)
            REDATA(I+1) = WAREA(3,I)
            WDATA(I+1) = WAREA(4,I)
27020    CONTINUE
C
         CALL  FFT ( REDATA, WDATA, M )
C
         DO  27040  I = 0,NPTS-1
            WAREA(3,I) = REDATA(I+1)
            WAREA(4,I) = WDATA(I+1)
27040    CONTINUE
C
         PRINT 98053
C
C
C
C      Implementing DECONV2 command.
C     -----------------------------
C
      ELSE IF ( FIELD(1) .EQ. 'DECONV2' )  THEN
C
         DO  27050  I = 0,NPTS-1
            TEMP = WAREA(1,I)*WAREA(1,I)+WAREA(2,I)*WAREA(2,I)
            Re = WAREA(3,I)
            Im = WAREA(4,I)
            WAREA(3,I) = (Re*WAREA(1,I)+Im*WAREA(2,I)) / TEMP
            WAREA(4,I) = (Im*WAREA(1,I)-Re*WAREA(2,I)) / TEMP
27050     CONTINUE
C
         PRINT 98054
C
C
C
C      Implementing DECONV3 command.
C     -----------------------------
C
      ELSE IF ( FIELD(1) .EQ. 'DECONV3' )  THEN
C
         DO  27060  I = 0,NPTS-1
            REDATA(I+1) = WAREA(3,I)
            WDATA(I+1) = WAREA(4,I)
27060    CONTINUE
C
         CALL  INVFFT ( REDATA, WDATA, M )
C
         DO  27070  I = 0,NPTS-1
            WAREA(1,I) = REDATA(I+1)
            WAREA(2,I) = WDATA(I+1)
27070    CONTINUE
C
         PRINT 98055
C
C
C
C
C
C    Implementing HELP command.
C    **************************
C
C
      ELSE IF ( FIELD(1) .EQ. 'HELP' )  THEN
C
```

```
            IF ( FIELD(2) .EQ. ' ' ) THEN
               PRINT 98061
            ELSE IF ( FIELD(2) .EQ. '1' ) THEN
               PRINT 98057
            ELSE IF ( FIELD(2) .EQ. '2' ) THEN
               PRINT 98058
            ELSE IF ( FIELD(2) .EQ. '3' ) THEN
            PRINT 98059
            ELSE IF ( FIELD(2) .EQ. '4' ) THEN
               PRINT 98060
            ELSE IF ( FIELD(2) .EQ. 'ALL' ) THEN
               PRINT 98057
               PAUSE 'Type CR for next page.'
               PRINT 98058
               PAUSE 'Type CR for next page.'
               PRINT 98059
               PAUSE 'Type CR for next page.'
               PRINT 98060
            ELSE
               PRINT 98061
            ENDIF
C
C
C
C
C
C
C
C    Implementing NOISE+ command.
C   ==============================
C
C
         ELSE IF ( FIELD(1) .EQ. 'NOISE+' ) THEN
C
C
            IF ( FIELD(2) .EQ. 'WA1' ) THEN
               WAINDX = 1
            ELSE IF ( FIELD(2) .EQ. 'WA2' ) THEN
               WAINDX = 2
            ELSE IF ( FIELD(2) .EQ. 'WA3' ) THEN
               WAINDX = 3
            ELSE IF ( FIELD(2) .EQ. 'WA4' ) THEN
               WAINDX = 4
            ELSE
               PRINT 99005
               WAINDX = 0
            ENDIF
C
            DECODE ( 98001, FIELD(3), ERR=28010 ) TEMP
            DECODE ( 98001, FIELD(4), ERR=28010 ) r
            DECODE ( 98001, FIELD(5), ERR=28010 ) REDATA(1)
C
            CALL  RANDN ( REDATA, NPTS, 0., TEMP )
C
            DO  28020  I = 0,NPTS-1
               WAREA(WAINDX,I) = WAREA(WAINDX,I) + r * REDATA(I+1)
28020       CONTINUE
            PRINT 98063, WAINDX
            GO TO 28030
C
28010       PRINT 99010
28030       CONTINUE
C
C
C
C
C
C    Implementing PLOT command.
C   ============================
C
C
         ELSE IF ( FIELD(1) .EQ. 'PLOT' ) THEN
C
C
            IF ( FIELD(2) .EQ. 'WA1' ) THEN
               WAINDX = 1
            ELSE IF ( FIELD(2) .EQ. 'WA2' ) THEN
               WAINDX = 2
            ELSE IF ( FIELD(2) .EQ. 'WA3' ) THEN
               WAINDX = 3
            ELSE IF ( FIELD(2) .EQ. 'WA4' ) THEN
               WAINDX = 4
            ELSE
               WAINDX = 0
               PRINT 99005
            ENDIF
C
C
           IF  ( WAINDX .NE. 0 ) THEN
C
              DO  29010  I = 0,NPTS-1
                 DATA(I) = WAREA(WAINDX,I)
29010         CONTINUE
C
```

```
            PRINT 98065
C
            PRINT 98066
            READ ( 8, 98006 ) COMAND
            CALL LINFLD ( COMAND, CLNGTH, FIELD, FLNGTH, NFLDS, ERRFLG )
            XLABEL = FIELD(1)
C
            PRINT 98067
            READ ( 8, 98006 ) COMAND
            CALL LINFLD ( COMAND, CLNGTH, FIELD, FLNGTH, NFLDS, ERRFLG )
            YLABEL = FIELD(1)
C
            PRINT 98068
            READ ( 8, 98006 ) COMAND
            CALL LINFLD ( COMAND, CLNGTH, FIELD, FLNGTH, NFLDS, ERRFLG )
            PTITLE = FIELD(1)
C
            PRINT 98069
            READ 98001, XAXIS, YAXIS
C
            PRINT 98070
            READ 98001, XMAX, XMIN, YMAX, YMIN
C
            PRINT 98071
            READ 98072, GDPNAM
C
C
C           Initiate plot.
C           --------------
C
            CALL NEWPAG
            CALL PAGNAM ( GDPNAM )
            CALL SLICIT ( AXIS, DATA, XAXIS, YAXIS, XMIN, XMAX, YMIN,
     &      YMAX, XLABEL, 20, YLABEL, 20, PTITLE, 20, NPTS )
C
            PRINT 98073, WAINDX
C
         ENDIF
C
C
C
C
C
C     Implementing BAYDECRS command.
C     ==============================
C
C
       ELSE IF ( FIELD(1) .EQ. 'BAYDECRS' )  THEN
C
          DECODE ( 98001, FIELD(2) ) NTIMES
C
C         Setting g=y.
C         ------------
C
          DO  31010  I = 1,NPTS-1
             WAREA(3,I) = WAREA(2,I)
31010     CONTINUE
C
C
          DO  31020  N = 1,NTIMES
C
C         Calculating p=x*g.
C         ------------------
C
             DO  31030  I = 0,NPTS-1
                PRSA(I)=0.
                DO  31040  J = 0,NPTS-1
                   SHIFT = I - J
                   IF ( SHIFT .LT. 0 )  SHIFT = SHIFT+NPTS
                   PRSA(I) = PRSA(I) + WAREA(1,J)*WAREA(3,SHIFT)
31040           CONTINUE
31030        CONTINUE
C
C         Calculating s=y/p.
C         ------------------
C
             DO  31050  I = 0,NPTS-1
                SRSA(I) = WAREA(2,I) / PRSA(I)
31050        CONTINUE
C
C         Calculating p=x*s.
C         ------------------
C
             DO  31060  I = 0,NPTS-1
                PRSA(I) = 0.
                DO  31070  J = 0,NPTS-1
                   SHIFT = MOD(I+J,NPTS)
                   PRSA(I) = PRSA(I) + WAREA(1,J)*SRSA(SHIFT)
31070           CONTINUE
31060        CONTINUE
C
```

```
C       Calculating g=g.p
C       -----------------
C
           DO  31080  I = 0,NPTS-1
              WAREA(3,I) = WAREA(3,I) * PRSA(1)
31080      CONTINUE
C
31020    CONTINUE
C
         PRINT 98075
         PRINT *,'   using real space approach.'
C
C
C
C
C
C
C     Implementing CORREL command.
C     ============================
C
C
      ELSE IF ( FIELD(1) .EQ. 'CORREL' )  THEN
C
         DO  32010  I = 0,NPTS-1
            WAREA(3,I) = 0.
            DO  32020  J = 0,NPTS-1
               SHIFT = MOD(I+J,NPTS)
               WAREA(3,I) = WAREA(3,I) + WAREA(1,J)*WAREA(2,SHIFT)
32020       CONTINUE
32010    CONTINUE
C
         PRINT  98085
C
C
C
C
C     Implementing SUM command.
C     =========================
C
C
      ELSE IF ( FIELD(1) .EQ. 'SUM' )  THEN
C
         IF ( FIELD(2) .EQ. 'WA1' )  THEN
            WAINDX = 1
         ELSE IF ( FIELD(2) .EQ. 'WA2' )  THEN
            WAINDX = 2
         ELSE IF ( FIELD(2) .EQ. 'WA3' )  THEN
            WAINDX = 3
         ELSE IF ( FIELD(2) .EQ. 'WA4' )  THEN
            WAINDX = 4
         ELSE
            PRINT 99005
            WAINDX = 0
         ENDIF
C
C
         IF ( WAINDX .NE. 0 )  THEN
C
            TEMP = 0.
            DO  33010  I = 0,NPTS-1
               TEMP = TEMP + WAREA(WAINDX,I)
33010       CONTINUE
            PRINT  98087, WAINDX, TEMP
C
         ENDIF
C
C
C
C
C
C    Implementing STOP command.
C    ==========================
C
C
      ELSE IF ( FIELD(1) .EQ. 'STOP' )  THEN
C
C
         PRINT 98007
         STOP
C
C
C
C
C
C
C   Printing error message if command is unrecognizable.
C   ====================================================
C
C
      ELSE
C
         PRINT 99004
C
      ENDIF
```

```
C
C
C
C   Going back for next command.
C   ============================
C
C
      GO TO 90001
C
C
C
C
C
C
C
C
C       F O R M A T    S T A T E M E N T S .
C      ======================================
C
C
C
98001 FORMAT ( )
C
98002 FORMAT ( 8(3X,F12.6) )
C
98003 FORMAT ( 1H1,/,3X,'WAVEPACK   V1.3     12/6/81',///)
C
98004 FORMAT ( /,3X,'Type HELP to list commands')
C
98005 FORMAT ( 3X,'ENTER COMMAND')
C
98006 FORMAT ( A80 )
C
98007 FORMAT ( /,3X,'WAVEPACK  TERMINATED     goodbye! ',///)
C
98008 FORMAT ( /,3X,'WA',I1,'  saved in store number  ',I1,/)
C
98009 FORMAT ( /,3X,'WA',I1,'  returned from store number  ',I1,/)
C
98010 FORMAT ( /,3X,'WA',I1,' label is  ',A12 )
C
C
98012 FORMAT ( /,3X,'Store ',I1,'  has label  ',A12 )
C
98013 FORMAT ( /,5X,'LABEL LISTS ARE:',/,4X,18('='),// )
C
98014 FORMAT ( /,3X,'WA',I1,'  stored in  ',A12 )
C
98015 FORMAT ( /,3X,'WA',I1,'  recalled from  ',A12 )
C
98016 FORMAT ( /,3X,'WA',I1,'  stored in store  ',I2 )
C
98017 FORMAT ( /,3X,'WA',I1,'  recalled from store  ',I2 )
C
98018 FORMAT ( /,3X,'FFT computed with Re = WA1 and Im = WA2' )
C
98019 FORMAT ( /,3X,'INVFFT computed with Re = WA1 and Im = WA2' )
C
98020 FORMAT ( /,3X,'Window codes are :',
     &         /,5X,'1 - Zero order window.',
     &         /,5X,'2 - First order window.',
     &         /,5X,'3 - Hanning window.',
     &         /,5X,'4 - Hamming window.',
     &         /,5X,'5 - Blackman window.',
     &        //,3X,'ENTER  window code,window centre  >>>')
C
98021 FORMAT ( /,3X,'Window number ',I1,' applied to WA',I1 )
C
98022 FORMAT ( /,3X,'Rectangular to polar conversion carried out.' )
C
98023 FORMAT ( /,3X,'Polar to rectangular conversion carried out.' )
C
98024 FORMAT ( /,3X,'Convolution of  WA3 = WA1 * WA2 carried out.' )
C
98025 FORMAT ( /,3X,'WA',I1,' Folded about the central point.' )
C
98026 FORMAT ( /,3X,'WA',I1,' set equal to ',F12.6,' from ',
     &         I3,' to ',I3,' inclusive.' )
C
98027 FORMAT ( /,3X,'WA1 and WA2 values have been exchanged.' )
C
98029 FORMAT ( /,3X,'WA',I1,' normalized by factor = ',F24.6 )
C
98030 FORMAT ( /,3X,'Sine transform applied to WA',I1 )
C
98031 FORMAT ( /,3X,'Arcsine transform applied to WA',I1 )
C
98032 FORMAT ( /,3X,'WA',I1,' rotated ',I3,' positions right.' )
C
98033 FORMAT ( /,3X,'WA',I1,' rotated ',I3,' positions left.' )
C
C
98035 FORMAT ( /,3X,'WA',I1,' set equal to WA',I1 )
C
```

```
98036 FORMAT( /,3X,'WA1 set equal to WA2  from ',I3,
     &        ' to ',I3,' inclusive. ' )
C
98037 FORMAT ( /,3X,'WA2 set equal to WA1 from ',I3,
     &         ' to ',I3,' inclusive. ' )
C
98038 FORMAT ( /,3X,'WA1 = WA1 * WA2' )
C
98039 FORMAT ( /,3X,'WA1 = WA1 / WA2' )
C
98040 FORMAT ( /,3X,'WA1 = WA2 / WA1' )
C
98041 FORMAT ( /,3X,'WA1 = WA1 + WA2' )
C
98042 FORMAT ( /,3X,'WA1 = WA1 - WA2' )
C
98043 FORMAT ( /,3X,'WA1 = WA2 - WA1' )
C
98044 FORMAT ( /,3X,'WA1 = WA1 *',F12.6 )
C
98045 FORMAT ( /,3X,'WA1 = WA1 /',F12.6 )
C
98046 FORMAT ( /,3X,'WA1 = WA1 +',F12.6 )
C
98047 FORMAT ( /,3X,'WA1 = WA1 -',F12.6 )
C
98048 FORMAT ( /,3X,'WA1 = 1 / WA1' )
C
98049 FORMAT ( /,3X,'SUBSTITUTE MODE - type E to exit.' )
C
98050 FORMAT ( /,3X,A12,'(',I3,') =',F12.6 )
C
98051 FORMAT ( /,3X,'SUBSTITUTE MODE  terminated.' )
C
98053 FORMAT ( /,3X,'WA1 = Re <F(x)>        WA2 = Im <F(x)>',
     &         /,3X,'WA3 = Re <F(y)>        WA4 = Im <F(y)>')
C
98054 FORMAT ( /,3X,'WA3 = Re <F(y)/F(x)>',8X,'WA4 = Im <F(y)/F(x)>')
C
98055 FORMAT ( /,3X,'WA1 = Re < IF[F(y)/F(x)] >',
     &           6X,'WA2 = Im < IF[F(y)/F(x)] >')
C
C
98057 FORMAT ( /,3X,'The WAVEPACK commands are :',/,2X,28('-'),/,
     & /,3X,'xxx - WA1, WA2, WA3 or WA4 only.',
     & /,3X,'yyy - WA1 or WA2 only.',
     & /,3X,'A - an integer 0 <= A <= NPTS-1',
     & /,3X,'B - an integer A <= B <= NPTS-1',
     & /,3X,'N - an integer, store number',
     & /,3X,'C - any real number.',
     &//,5X,'SAVE,N,xxx - stores work area xxx in ',
     &      'Fortran file N+10',
     & /,5X,'RETN,N,xxx - recalls work area xxx from ',
     &      'Fortran file N+10',
     & /,5X,'STO,N,xxx - store work area xxx in temporary ',
     &      'store N',
     & /,5X,'RCL,N,xxx - recall work area xxx from temporary ',
     &      'store N',
     &//,5X,'LLABEL - lists all the user defined labels',
     & /,5X,'SLABEL,xxx,label - sets label of work area xxx',
     & /,5X,'SLABEL,N - sets label of store number N',
     & /,5X,'RLABEL,xxx - prints current label of work area xxx',
     & /,5X,'RLABEL,N - prints current label of store N' )
C
98058 FORMAT ( //,5X,'PRINT,xxx - prints data in work area '
     &  ,'on screen',
     & /,5X,'PRINTF,xxx - prints data in work area xxx into ',
     &   'printfile',
     & /,5X,'GRAPH,xxx - graphs data in work area in work area ',
     &   'xxx on screen',
     & /,5X,'GRAPHF,xxx - graphs data in work area xxx into '
     &  ,'printfile',
     &//,5X,'FFT - computes the Fast Fourier Transform with',
     & /,5X,'      WA1 = Re<x> ,  WA2 = Im<x>',
     & /,5X,'INVFFT - computes the Inverse FFT data as for FFT',
     & /,5X,'WINDOW,xxx - applies window to work area xxx',
     & /,5X,'R-P - rectangular to polar conversion',
     & /,5X,'      WA1=Re, WA2=Im -> WA1=r, WA2=O',
     & /,5X,'P-R - polar to rectangular conversion',
     & /,5X,'      WA1=r, WA2=0 -> WA1=Re, WA2=Im',
     & /,5X,'CONV - convolves WA1 and WA2 result in WA3',
     & /,5X,'DECONV1 - computes first stage of deconvolution',
     & /,5X,'DECONV2 - computes second stage of deconvolution',
     & /,5X,'DECONV3 - computes third stage of deconvolution')
C
98059 FORMAT ( /,5X,'EXCH - exchanges the contents of WA1 '
     &          ,'and WA2',
     & /,5X,'FILL,xxx,C,A,B - sets work area xxx equal to C '
     &     ,'from A to B',
```

```
     & /,5X,'NORM,xxx - normalizes work area xxx',
     & /,5X,'SINE,xxx - sine transforms work area xxx',
     & /,5X,'ARCSINE,xxx - transforms work area xxx using '
     &     ,'arcsine transform',
     & /,5X,'ROTATE,xxx,A - rotates work area xxx A times',
     & //,3X,'ARITHMETIC COMMANDS:',
     & /,5X,'WA1=xxx - sets WA1 equal to work area xxx',
     & /,5X,'WA2=xxx -   + WA2   +   +   +   +   +  ',
     & /,5X,'WA3=xxx -   + WA3   +   +   +   +   +  ',
     & /,5X,'WA4=xxx -   + WA4   +   +   +   +   +  ',
     & /,5X,'SWA1=WA2,A,B - equates WA1 to WA2 from A to B',
     & /,5X,'SWA2=WA1,A,B -    +    WA2  + WA1  +   +   + +  ',
     & /,5X,'WA1=WA1+WA2   WA1=WA1/WA2   WA1=WA2/WA1',
     & /,5X,'WA1=WA1+WA2   WA1=WA1-WA2   WA1=WA2-WA1',
     & /,5X,'WA1=WA1*,C    WA1=WA1/,C    WA1=WA1+,C',
     & /,5X,'WA1=WA1-,C    WA1=1/WA1' )
C
98060 FORMAT ( /,5X,'SUBST,xxx,A - enter substitute mode '
     &     ,'with xxx at element A',
     & /,5X,'STOP - terminates WAVEPACK',
     & /,5X,'HELP,A - prints these commands or page A only' ,
     & /,5X,'NOISE+,xxx,stdev,amplit,seed - adds noise to xxx'
     & /,5X,'SUM,xxx - computes the summation of all the data '
     &                 ,'elements of work area xxx.'
     & /,5X,'BAYDECRS,N - implements Bayesian deconvolution N times'
     & /,5X,'CORREL - computes crosscorrelation of WA1 and WA2',
     &              ' result in WA3.' )
C
98061 FORMAT ( /,3X,'Index to commands :',/,1X,22('-'),
     & /,5X,'page 1 -- storing and recalling data,label commands',
     & /,5X,'page 2 -- displaying results,signal processing',
     & /,5X,'page 3 -- sine transforms,arithmetic commands',
     & /,5X,'page 4 -- substitute,help,stop commands',
     & //,5X,'TYPE - HELP,A  where A = desired page number',/)
C
98063 FORMAT ( /,3X,'Noise added to WA',I1 )
C
98075 FORMAT ( /,3X,'Bayesian deconvolution carried out' )
C
98065 FORMAT ( /,3X,'GDP plotting interface.' )
C
98066 FORMAT ( /,3X,'ENTER >>  xaxis-label' )
C
98067 FORMAT ( /,3X,'ENTER >>  yaxis-label' )
C
98068 FORMAT ( /,3X,'ENTER >>  plot-title' )
C
98069 FORMAT ( /,3X,'ENTER >>  axes length in cms. xaxis-length'
     &            ,',yaxis-length' )
C
98070 FORMAT ( /,3X,'ENTER >>  xmax,xmin,ymax,ymin' )
C
98071 FORMAT ( /,3X,'ENTER >>  GDP page name for this plot' )
C
98072 FORMAT ( A12 )
C
98073 FORMAT ( /,3X,'WA',I1,' plotted.' )
C
98085 FORMAT ( /,3X,'Correlation of WA3 = WA1 & WA2 carried out.')
C
98087 FORMAT ( /,3X,'Sum of all the elements of WA',I1,' = ',F12.6)
C
C
C
C
C
C
99001 FORMAT ( /,3X,'*** ERROR - contents of field 3 is',
     &         ' unrecognizable ***',/ )
C
99002 FORMAT ( /,3x,'*** ERROR - contents of field 2 out',
     &         ' of range ***',/ )
C
99003 FORMAT ( /,3X,'*** ERROR -  command syntax ***',/)
C
99004 FORMAT ( /,3X,'*** ERROR - unrecognizable command ***',/)
C
99005 FORMAT ( /,3X,'***ERROR - contents of field 2 ',
     &         'unrecognizable ***',/ )
C
99006 FORMAT ( /,3X,'*** ERROR - contents of field 1 '
     &            ,'unrecognizable ***' )
C
99007 FORMAT ( /,3X,'*** ERROR - contents of field 4 ',
     &            'unrecognizable ***',/ )
C
99008 FORMAT ( /,3X,'*** ERROR - contents of field 5 ',
     &             'unrecognizable ***',/ )
C
99009 FORMAT ( /,3X,'*** ERROR - contents of field 3 ',
     &            'out of range ***' )
C
99010 FORMAT ( /,3X,'*** ERROR - noise command error ***',/)
C
      END
END OF FILE
```

```
C  TITLE:  LINFLD
C
C
C  DESCRIPTION:
C
C  This subroutine accepts as input a character variable containing
C  the input card just read.  LINFLD then breaks up the input  card
C  into fields separated by commas and returns these fields to  the
C  calling program.  Termination of input  occurs  when  the  first
C  blank in the input string is detected.  An error flag is availa-
C  ble if needed.  All the fields are cleared and  the  error  flag
C  is set or cleared.
C  The call line is :
C
C     CALL LINFLD ( LINEIN, LLNGHT, FLDOUT, FLNGTH, NFLDS, ERRFLG )
C
C  The parameters are :
C
C   LINEIN - a character variable containing the input record (input).
C
C   LLNGTH - an integer specifying the number of characters in LINEIN
C            must be set to 80 in the main program (input).
C
C   FLDOUT - a character array of dimension NFLDS (input).
C
C   FLNGTH - an integer specifying the maximum number of alphanumeric
C            characters in each FLDOUT character variable.  Must be
C            set to 20 in the main program (input).
C
C   NFLDS - an integer specifying the number of character variables
C           in the FLDOUT character array, probably set as a parameter
C           in the main program (input).
C
C   ERRFLG - an integer set by the subroutine :
C                                              0 = no error
C                                              1 = error
C
C            An error may arise if the specified field length or line
C            length is exceeded.
C
C      In the main program the required definitions are:
C
C       CHARACTER  LINEIN*80,  FLDOUT*20 ( 1:NFLDS )
C
C  CALLS :   nothing
C
C
C  CODE :
C
      SUBROUTINE  LINFLD ( LINEIN, LLNGTH, FLDOUT, FLNGTH, NFLDS,
     &                     ERRFLG )
C
C    Defining the variables used.
C    ============================
C
      INTEGER  LLNGTH, FLNGTH, NFLDS, ERRFLG
      INTEGER  CCOUNT, FCOUNT, FLDNO, I, J
C
      CHARACTER  LINEIN*80, FLDOUT*20 (1:NFLDS)
C
C
C    Initializing variables.
C    =======================
C
      ERRFLG = 0
      DO 100  I=1,NFLDS
         FLDOUT(I) = 20H
100   CONTINUE
C
      FLDNO = 1
      CCOUNT = 1
      FCOUNT = 1
C
```

```
C
C    Sorting the fields from the input record.
C    =========================================
C
55     IF  ( LINEIN(CCOUNT:CCOUNT) .EQ. ' ' )  THEN
          RETURN
C
       ELSE
          IF  ( LINEIN(CCOUNT:CCOUNT) .EQ. ',' )  THEN
             FCOUNT = 1
             FLDNO = FLDNO+1
             CCOUNT = CCOUNT +1
C
             IF  ( FLDNO .GT. NFLDS )  THEN
                ERRFLG = 1
                RETURN
             ELSE
                GO TO 55
             ENDIF
C
          ELSE
             FLDOUT(FLDNO)(FCOUNT:FCOUNT) = LINEIN(CCOUNT:CCOUNT)
             FCOUNT = FCOUNT+1
             CCOUNT = CCOUNT+1
C
             IF  ( FCOUNT .GT. FLNGTH .OR. CCOUNT .GT. LLNGTH )  THEN
                ERRFLG = 2
                RETURN
             ELSE
                GO TO 55
             ENDIF
C
          ENDIF
C
       ENDIF
C
C
       END
END OF FILE
->


C  TITLE :  GRAPHW
C
C  DESCRIPTION :
C
C  THIS SUBROUTINE IS USED AS A DRIVER ROUTINE FOR THE
C  GRAPHV SUBROUTINE WHICH PRINTS  A  CHARACTER  GRAPH
C  ON THE VDU SCREEN. THIS  ROUTINE  ACCEPTS  AS INPUT
C  THE DATA ARRAY TO BE PLOTTED AND  NORMALISES IT BE-
C  FORE PLOTTING. IT CALLS GRAPHV.
C
C
C  CALLS :  GRAPHV
C
C  THE CALL LINE IS :
C
C      CALL GRAPHW ( AXIS, DATA, YEXP, XEXP, NPTS, DEVICE )
C
C  PARAMETERS :
C
C  AXIS - A REAL ARRAY OF DIMENSION NPTS CONTAINING THE
C           X AXIS VALUES.
C
C  DATA - A REAL ARRAY OF DIMENSION NPTS CONTAINING THE
C         Y AXIS VALUES.
C
C  YEXP - A 12 CHARACTER TITLE.
C
C  XEXP - A 72 CHARACTER TITLE.
C
C  NPTS - AN INTEGER SPECIFYING THE NUMBER OF POINTS TO
C         BE PLOTTED.
C
C  DEVICE - AN INTEGER SPECIFYING THE DEVICE TO WHICH
C           THE OUTPUT IS TO BE SENT .
C
C  CODE :
C
      SUBROUTINE  GRAPHW ( AXIS, DATA, YEXP, XEXP, NPTS, DEVICE )
C
C
C  DEFINING THE VARIABLES TO BE USED
C ==================================
C
      IMPLICIT  COMPLEX (A - Z)
C
      INTEGER  NPTS, I, DEVICE
C
      REAL  AXIS (0:NPTS-1), DATA (0:NPTS-1)
      REAL  TOP, BOTTOM
      REAL  MAXX, MINX, MAXY, MINY
C
      CHARACTER YEXP * 12, XEXP * 72
```

```
C
C  COMPUTING THE LIMITS OF THE GRAPH
C ==================================
C
C
      MAXX = AXIS (0)
      MINX = AXIS (0)
      MAXY = DATA (0)
      MINY =  0.
C
      DO 100  I = 0,NPTS-1
        IF ( DATA (I) .GT. MAXY )  MAXY = DATA (I)
        IF ( DATA (I) .LT. MINY )  MINY = DATA (I)
        IF ( AXIS (I) .GT. MAXX )  MAXX = AXIS (I)
        IF ( AXIS (I) .LT. MINX )  MINX = AXIS (I)
100   CONTINUE
C
C
C  SETTING THE GRAPH LIMITS
C ========================
C
      TOP = ( MAXY - MINY ) * .1 + MAXY
      BOTTOM = MINY - (MAXY - MINY) * .1
C
C
      CALL GRAPHV ( AXIS, DATA, YEXP, XEXP, MINX,
     &              MAXX, BOTTOM, TOP, NPTS, ' ', DEVICE )
C
      RETURN
      END
END OF FILE
->
```

```
C  TITLE :   GRAPHV
C
C  DESCRIPTION :
C
C  THIS SUBROUTINE PRINTS A CHARACTER GRAPH ON A STANDARD
C  24 LINE VDU SCREEN.  THE USER HAS THE CHOICE OF SPECI-
C  FYING THE CHARACTERS WHICH WILL MAKE  UP  THE  PRINTED
C  CURVE OR ALLOWING THE PROGRAM TO IMPLEMENT A  FORM  OF
C  BEST CHARACTER FIT USING  . , *  OR '
C
C  THE CALL LINE IS :
C
C   CALL GRAPHV ( X, Y, YEXP, XEXP, XMIN, XMAX, YMIN,
C                 YMAX, NPTS, CHAR )
C
C
C  PARAMETERS :
C
C  X - A REAL ARRAY OF DIMENSION 'NPTS' CONTAINING THE X-
C      COORDINATES  OF  ALL  THE  POINTS  TO  BE PLOTTED.
C      ARRAY DIMENSION SHOULD BE ( 0:NPTS-1 ).
C
C  Y - A REAL ARRAY OF DIMENSION 'NPTS' CONTAINING THE Y-
C      COORDINATES  OF  ALL  THE  POINTS  TO  BE PLOTTED.
C      ARRAY DIMENSION SHOULD BE ( 0:NPTS-1 ).
C
C  YEXP - A STRING OF 12 CHARACTERS IN LENGTH WHICH  WILL
C         BE PRINTED AS THE Y-AXIS TITLE.
C
C  XEXP - A STRING OF 72 CHARACTERS IN LENGTH WHICH  WILL
C         BE PRINTED AS THE X-AXIS TITLE.
C
C  XMIN - A REAL VALUE, THE SMALLEST VALUE  OF  X  TO  BE
C         PLOTTED.
C
C  XMAX - A REAL VALUE, THE LARGEST  VALUE  OF  X  TO  BE
C         PLOTTED.
C
C  YMIN - A REAL VALUE, THE SMALLEST VALUE  OF  Y  TO  BE
C         PLOTTED.
C
C  YMAX - A REAL VALUE, THE LARGEST  VALUE  OF  Y  TO  BE
C         PLOTTED.
C
C  NPTS - AN INTEGER, THE TOTAL NUMBER OF  POINTS  TO  BE
C         PLOTTED.
C
C  CHAR - A CHARACTER STRING, ONE  CHARACTER  IN  LENGTH.
C         THIS INDICATES TO 'GRAPHV' WHICH TYPE OF CHARA-
C         TERS ARE TO BE USED TO PRINT THE CURVE.
C         IF :
C               CHAR = 'any character'  - THE   SPECIFIED
C                                         CHARACTER    IS
C                                         USED.
C               CHAR = 'blank' - EITHER A  . * '  WILL  BE
C                                USED TO ACHIEVE  A  BEST
C                                FIT.
C  DEVICE - AN INTEGER SPECIFYING THE DEVICE TO WHICH THE
C           PRINTOUT IS TO BE SENT (USUAL FORTRAN SPECS).
C
```

```
C
C
C  CODE :
C
      SUBROUTINE  GRAPHV ( X, Y, YEXP, XEXP, XMIN, XMAX,
     &                     YMIN, YMAX, NPTS, CHAR, DEVICE )
C
C
C
C  DEFINING THE VARIABLES TO BE USED.
C ===================================
C
C
      IMPLICIT  COMPLEX (A - Z)
C
      PARAMETER  XSTEP = 14, YSTEP = 5
      PARAMETER  WIDTH = 4 * XSTEP + 1, LENGTH = 4 * YSTEP + 1
C
      INTEGER  I, J, NPTS
      INTEGER  XCOORD, YCOORD, YTEMP, DEVICE
C
      REAL  X (0:NPTS-1), Y (0:NPTS-1)
      REAL  XMIN, XMAX, YMIN, YMAX
      REAL  XLABEL (5)
C
C
      CHARACTER * 1  GRAPH ( LENGTH, WIDTH )
      CHARACTER * 12  YLABEL ( LENGTH )
      CHARACTER  YEXP * 12, XEXP * 72
      CHARACTER * 1  CHAR, PCHAR
      CHARACTER * 1  APPROX ( 3 )
C
C
C
C  IN THE ABOVE DEFINITION BLOCK THE INPUT PARAMETERS
C  ARE DEFINED ALONG WITH SEVERAL INTERNAL  VARIABLES
C  AND INTERNAL PARAMETERS.
C
C  LOCAL VARIABLES AND PARAMETERS :
C
C  XSTEP - THE NUMBER OF  PRINT  CHARACTERS  BETWEEN
C          THE X DISTANCE MARKERS.
C
C  YSTEP - THE NUMBER OF  PRINT  CHARACTERS  BETWEEN
C          THE Y DISTANCE MARKERS.
C
C  WIDTH - GRAPH WIDTH AS DEFINED IN TERMS OF  XSTEP
C
C  LENGTH - GRAPH LENGTH DEFINED IN TERMS  OF  YSTEP
C
C  I , J  - DO LOOP RUNNING VARIABLES
C  XCOORD , YCOORD  - ARRAY  INDICES  FOR  THE  CAR-
C                     TESIAN TYPE COORDINATES OF THE
C                     POINT UNDER CONSIDERATION
C
C  YTEMP - TEMPORARY INTEGER VALUE
C
C  XLABEL - THE FIVE DISTANCE MARKERS TO BE  PRINTED
C           FOR THE X - AXIS .
C
C
C  CHARACTERS :
C
C  GRAPH - AN ARRAY IN WHICH THE CURVE AND ITS  AXES
C          IS GENERATED
C
C  YLABEL - AN ARRAY WHICH CONTAINS THE Y  AXIS MAR-
C           KERS AND LABEL
C
C  PCHAR - THE  ACTUAL  CHARACTER  REPRESENTING  THE
C          CURVE TO BE PRINTED.
C
C  APPROX - A 3-DIMENSIONAL ARRAY CONTAINING . * '
C
C
C
C
```

```
C
C  INITIALIZING THE VARIABLES TO BE USED
C ======================================
C
C
      APPROX ( 1 ) = '.'
      APPROX ( 2 ) = '*'
      APPROX ( 3 ) = 1H'
C
      DO 100  I = 1, LENGTH
      YLABEL (I) = 12H
C
      DO 200  J = 1, WIDTH
      GRAPH (I, J) = ' '
200   CONTINUE
C
100   CONTINUE
C
C
C
C  DRAWING THE AXES ONTO THE GRAPH
C ===============================
C
C
      DO 300  I = 1, WIDTH
      GRAPH (1, I) = '-'
      GRAPH (LENGTH, I) = '-'
300   CONTINUE
C
C
      DO 400  I = 1, LENGTH
         GRAPH (I, 1) = 'I'
         GRAPH (I, WIDTH) = 'I'
400   CONTINUE
C
C
      DO 500  I = 1, LENGTH, YSTEP
C
      DO 600  J = 1, WIDTH, XSTEP
      GRAPH (I, J) = '+'
600   CONTINUE
C
500   CONTINUE
C
C
C
C
C  SETTING UP AXIS LABLES AND MARKERS
C ==================================
C
C
      YLABEL (LENGTH / 2 - 2) = YEXP
C
      DO 700  I = 1,5
      XLABEL (I) = XMIN + (XMAX - XMIN) * (I - 1) / 4
      ENCODE (1000, YLABEL ( YSTEP * (I - 1) + 1) )
     &       YMIN + (YMAX - YMIN) * (5 - I) / 4
700   CONTINUE
C
C
C
C
C  ENTERING THE CURVE CHARACTERS INTO THE GRAPH ARRAY
C ================================================
C
C
      DO 800  I = 0,NPTS-1
C
      PCHAR = CHAR
      XCOORD = INT ( WIDTH * ( X(I) - XMIN ) / ( XMAX - XMIN ) ) + 1
C
C  TESTING TO DETERMINE IF THE X COORDINATE IS WITHIN
C  THE MAXIMUM AND MINIMUM VALUES. IF NOT IT  IS  SET
C  EQUAL TO THE CORRESPONDING LIMIT.
C
      IF  ( XCOORD .GT. WIDTH )  XCOORD = WIDTH
      IF  ( XCOORD .LT. 1 )  XCOORD = 1
C
C  TESTING TO DETERMINE IF THE Y COORDINATE IS WITHIN
C  THE MAXIMUM AND MINIMUM VALUES. IF NOT IT  IS  SET
C  EQUAL TO THE CORRESPONDING LIMIT.
C
      YTEMP = INT ( 3 * LENGTH * ( Y(I) - YMIN ) / ( YMAX - YMIN ) )
C
      IF  ( YTEMP .GT. 3 * LENGTH - 1 )  YTEMP = 3 * LENGTH - 1
      IF  ( YTEMP .LT. 0 ) YTEMP = 0
C
```

```
C
C  TESTING IF THE 'BEST FIT' MUST BE APPLIED AND IF SO
C  PRINTING THE CHARACTER WHICH APPROXIMATES THE CURVE
C  BEST.
C
      IF  ( CHAR .EQ. ' ' ) PCHAR = APPROX ( MOD (YTEMP, 3) + 1 )
C
      YCOORD = YTEMP / 3 + 1
C
C  ACTUALLY PLACING THE CHARACTER ON THE GRAPH
C
      GRAPH ( YCOORD, XCOORD ) = PCHAR
C
800   CONTINUE
C
C
C
C
C  PRINTING THE GRAPH ON THE VDU SCREEN
C ====================================
C
C
      WRITE ( DEVICE, 5000 )
C
      DO 900  I = 1, LENGTH
      WRITE ( DEVICE, 2000 )  YLABEL (I), ( GRAPH (LENGTH+1-I, J),
     &                                       J = 1,WIDTH )
900   CONTINUE
C
C
      WRITE ( DEVICE, 3000 )  XLABEL
C     WRITE ( DEVICE, 4000 )  XEXP
C
      RETURN
C
C
C
C  FORMAT STATEMENTS
C ==================
C
1000  FORMAT ( 4X, F8.2 )
2000  FORMAT ( 2X, A12, 2X, 70A1 )
3000  FORMAT ( 5X, 5(6X, F8.2) )
4000  FORMAT ( A72 )
5000  FORMAT (1H1)
C
C
      END
END OF FILE
->



C  TITLE :      PRINT    V1.0
C
C  DESCRIPTION :
C
C  THIS SUBROUTINE PRINTS OUT NPTS OF DATA ALONG WITH
C  THE INDEX ROW BY ROW. THE DATA IS PRESENTED  BELOW
C  ITS RESPECTIVE INDEX AT 8 PER LINE.
C
C  PARAMETERS :
C
C       NPTS - NUMBER OF DATA POINTS IN ARRAY,
C              MUST BE A POWER OF TWO.
C
C       DATA - ARRAY CONTAINING THE REVELENT DATA.
C
C       TITLE - A 12 CHARACTER WHICH IS USED AS A HEADING
C
C      DEVICE - AN INTEGER SPECIFYING TO WHICH DEVICE PRINTOUT
C               IS TO BE SENT  ( USUAL FORTRAN SPECIFICATIONS).
C
C
C
C
C  CODE :
C
C     SUBROUTINE  PRINT (DATA, NPTS, TITLE, DEVICE)
C
C  INITIALIZING THE VARIABLES TO BE USED
C
      IMPLICIT COMPLEX (A - Z)
      INTEGER   NPTS, A, B, DEVICE
      REAL  DATA (0:NPTS-1)
      CHARACTER * 12  TITLE
C
```

```
C
C  PRINTING THE DATA ROW BY ROW
C
C
      WRITE ( DEVICE, 4000 )  TITLE
C
C
C
      DO 100  A = 0,NPTS-8,8
C
      WRITE ( DEVICE, 1000 )  ( B+A ,  B = 0, 7 )
      WRITE ( DEVICE, 2000)  ( DATA (A+B),  B = 0, 7 )
C
100   CONTINUE
C
C
C  PRINTING SOME BLANK LINES
C
      WRITE ( DEVICE, 3000 )
C
C  FORMAT STATEMENTS
C
1000  FORMAT ( ///,' DATA INDEX :   ', 3X, 8(I4, 6X) )
C
2000  FORMAT ( /,7X,'DATA :    ', 8(F10.6, 2X) )
C
3000  FORMAT ( // )
C
4000  FORMAT ( 1H1, ///, 40X, A12, 5X, 'DATA', /, 38X, 25('=') )
C
      RETURN
      END
END OF FILE
->
```

```
C***********************************************
C  THIS SUBROUTINE COMPUTES THE FAST FOURIER
C  TRANSFORM USING THE COOLEY-TUKEY ALGORITHM
C    [ RADIX 2 , DECIMATION IN TIME ]
C***********************************************

      SUBROUTINE FFT(X,Y,M)
      DIMENSION X(1),Y(1)
      INTEGER REP,DISP
      PI=3.1415927
      N=2**M
C
C**** THE FOLLOWING SECTION REORDERS THE INPUT DATA ****
C**** SO THAT THE TRANSFORM MAY BE COMPUTED IN PLACE ***
C
      NM1=N-1
      J=1
      DO 30 I=1,NM1
      IF(I.GE.J) GO TO 10
      T1=X(J)
      X(J)=X(I)
      X(I)=T1
      T2=Y(J)
      Y(J)=Y(I)
      Y(I)=T2
10    K=N/2
20    IF(K.GE.J) GO TO 30
      J=J-K
      K=K/2
      GO TO 20
30    J=J+K
C
C**** THE FOLLOWING CODE ACTUALLY COMPUTES THE FFT ****
C
      DO 40 I=1,M
      REP=2**I
      DISP=REP/2
      ARG=2*PI/REP
      DO 40 J=1,DISP
      TWF=(J-1)*ARG
      C=COS(TWF)
      S=SIN(TWF)
      DO 40 K1=J,N,REP
      J2=K1+DISP
      T1=C*X(J2)+S*Y(J2)
      T2=-S*X(J2)+C*Y(J2)
      X(J2)=X(K1)-T1
      Y(J2)=Y(K1)-T2
      X(K1)=X(K1)+T1
      Y(K1)=Y(K1)+T2
40    CONTINUE
C
      RETURN
      END
```

```
C
C
C  THE PROGRAM COMPUTES THE FFT OF THE DATA CONTAINED
C  IN THE X AND Y ARRAYS. THE X ARRAY CONTAINS THE
C  REAL PART OF THE DATA AND THE Y ARRAY THE IMAGINARY
C  PART. THUS A POINT < A ; B > IS ENTERED AS
C              X(..) = A
C              Y(..) = B
C  THE TOTAL NUMBER OF POINTS MUST BE A POWER OF 2
C  AND M IS THE POWER TO WHICH 2 MUST BE RAISED
C  I.E.  2**M
C
C
C*****************************************************************
C*          THIS SUBROUTINE COMPUTES THE INVERSE                 *
C*          FAST FOURIER TRANSFORM AND CALLS THE                 *
C*                  ABOVE SUBROUTINE                             *
C*****************************************************************
C
      SUBROUTINE INVFFT(X,Y,M)
      DIMENSION X(1),Y(1)
      N=2**M
      DO 10 I=1,N
      Y(I)=-Y(I)
10    CONTINUE

      CALL FFT(X,Y,M)

      DO 20 J=1,N
      X(J)=X(J)/N
20    CONTINUE

      RETURN
      END

END OF FILE
->



C  TITLE :  WINDOW
C
C
C  DESCRIPTION :
C
C  THIS SUBROUTINE APPLIES ONE OF FIVE WINDOWS TO THE
C  ARRAY SPECIFIED. THE INPUT AND OUTPUT  ARRAYS  MAY
C  BE THE SAME ARRAY.
C
C
C  THE CALL LINE IS :
C
C   CALL  WINDOW ( DATAIN, DATOUT, N, L, WIND )
C
C  PARAMETERS :
C
C     DATAIN - A REAL ARRAY OF DIMENSION N CONTAINING
C              THE INPUT DATA.
C
C     DATOUT - A REAL ARRAY OF DIMENSION N CONTAINING
C              THE WINDOWED DATA.
C
C          N - AN INTEGER SPECIFYING THE NUMBER OF DATA.
C
C          L - AN INTEGER SPECIFYING THE NUMBER OF DATA
C              POINTS IN THE WINDOW.
C
C       WIND - AN INTEGER WHICH SPECIFYS THE WINDOW TO BE
C              APPLIED TO THE DATA.
C
C            = 1  - ZERO ORDER WINDOW
C            = 2  - FIRST ORDER WINDOW
C            = 3  - HANNING WINDOW
C            = 4  - HAMMING WINDOW
C            = 5  - BLACKMAN WINDOW
C
C
C  CODE :
C
      SUBROUTINE  WINDOW ( DATAIN, DATOUT, N, L, WIND )
C
C  DEFINING THE VARIABLES USED
C ============================
C
      IMPLICIT COMPLEX (A - Z)
C
      INTEGER  N, L, WIND, I
C
      REAL  DATAIN (N), DATOUT (N)
      REAL  A1, A2
C
```

```
C
C  SETTING THE OUTPUT DATA EQUAL TO THE INPUT DATA INITIALLY
C ============================================================
C
      DO 600  I = 1, N
         DATOUT (I) = DATAIN (I)
600   CONTINUE
C
C
C  IMPLEMENTING THE DESIRED WINDOW
C ================================
C
      WIND = MAX0 ( 1, MIN0 ( 5, WIND ) )
      GO TO ( 10,20,30,40,50 )  WIND
C
C
C  ZERO ORDER WINDOW
C ==================
C
10    IF ( L .GE. N )  RETURN
C
      DO 100  I = L+1, N
         DATOUT (I) = 0.
100   CONTINUE
C
      RETURN
C
C
C  FIRST ORDER WINDOW
C ===================
C
20    A1 = (L - 1.) / 2.
C
      DO 200  I = 1, L
         W = 1. - ABS (I - 1. - A1) / A1
         DATOUT (I) = DATOUT (I) * W
200   CONTINUE
C
      GO TO 10
C
C
C  HANNING WINDOW
C ===============
C
30    A1 = 6.23818 / (L - 1.)
C
      DO 300  I = 1, L
         W = .5 - .5 * COS ( (I-1.)*A1 )
         DATOUT (I) = DATOUT (I) * W
300   CONTINUE
C
      GO TO 10
C
C
C  HAMMING WINDOW
C ================
C
40    A1 = 6.23818 / (L-1.)
C
      DO 400  I = 1, L
         W = .54 - .46 * COS ( (I-1.)*A1 )
         DATOUT (I) = DATOUT (I) * W
400   CONTINUE
C
      GO TO 10
C
C
C  BLACKMAN WINDOW
C ================
C
50    A1 = 6.23818 / (L-1.)
      A2 = 2. * A1
C
      DO 500  I = 1, L
         W = .42 - .5 * COS ( (I-1.)*A1 ) + .08 * COS ( (I-1.)*A2 )
         DATOUT (I) = DATOUT (I) * W
500   CONTINUE
C     GO TO 10
C
      END
END OF FILE
->
```

```
      SUBROUTINE  SLICIT(X,Y,XAXIS,YAXIS,XMIN,XMAX,YMIN,YMAX,XLABEL,NXCH
     +            AR,YLABEL,NYCHAR,TITLE,NTCHAR,NPNTS)
C
C
C        THIS ROUTINE PLOTS ONTO A NEW GDP PAGE A GRAPH OF THE
C     X POINTS VS THE Y POINTS.  IT ALSO PLACES A SCALE
C     ALONG THE TWO AXES & WRITES THE LABELS ALONG THE CORRECT
C     AXIS.
C
C
C        INPUTS
C        ======
C
C
C     X             ARRAY OF POINTS TO BE PLOTTED
C
C     Y             ARRAY OF Y VALUES TO BE PLOTTED AGAINST THE X VALUES
C
C     XAXIS         THE X AXIS LENGTH IN CMS
C
C     YAXIS         THE Y AXIS LENGTH IN CMS
C
C     XLABEL        THE X AXIS LABEL
C
C     NXCHAR        THE NUMBER OF CHARACTERS IN THE X AXIS LABEL
C
C     YLABEL        THE Y AXIS LABEL
C
C     NYCHAR        THE NUMBER OF CHARACTERS IN THE Y AXIS LABEL
C
C     TITLE         THE TITLE OF THE PLOT
C
C     NTCHAR        THE NUMBER OF CHARACTERS IN THE PLOT TITLE
C
C     NPNTS         THE NUMBER OF POINTS IN THE X AND Y ARRAYS
C
C     XMIN          MINIMUM X VALUE
C
C     XMAX          MAXIMUM X VALUE
C
C     YMIN          MINIMUM Y VALUE
C
C     YMAX          MAXIMUM Y VALUE
C
C        OUTPUTS
C        =======
C
C     THE ROUTINE OUTPUTS TO A NEW GDP PLOT FILE THE GRAPH.
C
C
C        LOCAL VARIABLES
C        ===============
C
C
C
C     XRANG         THE DIFFERENCE BETWEEN THE X VALUES AT THE
C                   ORIGIN AND AT THE END OF THE X AXIS
C
C     YRANG         THE DIFFERENCE BETWEEN THE Y VALUES AT THE
C                   ORIGIN AND AT THE END OF THE Y AXIS
C
C     XSTART        THE VALUE OF THE ORIGIN IN TERMS OF THE
C                   X VALUES
C
C     YSTART        THE VALUES OF THE ORIGIN IN TERMS OF THE
C                   Y VALUES
C
C     XVST          THE CHANGE IN VALUE BETWEEN TWO SUCCESSIVE X AXIS
C                   SCALE MARKS.
C
C     YVST          THE CHANGE IN VALUE BETWEEN TWO SUCCESSIVE Y AXIS
C                   SCALE MARKS.
C
C     XSTEP         THE NUMBER OF CMS BETWEEN SUCCESSIVE X AXIS SCALE
C                   MARKS
C
C     YSTEP         THE NUMBER OF CMS BETWEEN SUCCESSIVE Y AXIS SCALE
C                   MARKS
C
C     XCHAR         THE CALCULATED MAXIMUM HEIGHT OF X LABEL CHARACTERS
C
C     YCHAR         THE CALCULATED MAXIMUM HEIGHT OF Y LABEL CHARACTERS
C
C     CHARH         THE SIZE OF THE LABEL CHARACTERS IN CMS
C
C     NXLNS         THE CALCULATED NUMBER OF LINES NEEDED FOR THE
C                   XAXIS LABEL
C
C     NYLNS         THE CALCULATED NUMBER OF LINES NEEDED FOR THE
C                   Y AXIS LABEL
C
C     NTLNS         THE CALCULATED NUMBER OF LINES NEEDED FOR THE
C                   PLOT TITLE
```

```
C
C
C       DECLARATIONS
C       ============
C
C
      REAL  X(1), Y(1)
C
      CHARACTER*4  XLABEL(1), YLABEL(1), TITLE(1)
C
C
C     SET UP THE SCALE VALUES FOR THE LABLING
C
C     ---- GET THE RANGE OF VALUES ----
C
      XRANG = XMAX - XMIN
      YRANG = YMAX - YMIN
C
C     ---- GET FIRST VALUES ----
C
      XSTART = XMIN
      YSTART = YMIN
C
      XLAST = XMAX
      YLAST = YMAX
C
C     ---- WORK OUT HOW MANY TICKS TO HAVE ----
C
      NXTICS = IFIX(XAXIS/4.0 + 0.5) + 1
      NYTICS = IFIX(YAXIS/4.0 + 0.5) + 1
C
C     ---- GET THE DISTANCE BETWEEN TWO SCALE MARKS ----
C
      XSTEP = XAXIS/(NXTICS-1)
      YSTEP = YAXIS/(NYTICS-1)
C
C     ---- GET THE CHANGE OF VALUE BETWEEN TWO SCALE MARKS ----
C
      XVST = (XLAST-XSTART)/FLOAT (NXTICS-1)
      YVST = (YLAST-YSTART)/FLOAT (NYTICS-1)
C
C     ---- GET THE FACTORS TO CONVERT THE X & Y POINTS ----
C
      XFAC = XAXIS/XRANG
      YFAC = YAXIS/YRANG
C
C  WORK OUT THE PAGE SIZE NEEDED FOR THE PLOT
C
C
C     ---- WHAT SPACE FOR THE LABELS ----
C
      IF(NXCHAR .GT. 80) THEN
        XCHARH = XAXIS / 80.0
      ELSE
        XCHARH = 0.5
      ENDIF
C     IF(NTCHAR .GT. 80) THEN
        TCHARH = XAXIS / 80.0
      ELSE
        TCHARH = 0.5
      ENDIF
C
      IF(NYCHAR .GT. 80) THEN
        YCHARH = YAXIS / 80.0
      ELSE
        YCHARH = 0.5
      ENDIF
C
C     ---- SET THE NUMBER OF LINES OF EACH AXIS LABEL ----
C
C
      NXLNS = 1
      NYLNS = 1
      NTLNS = 1
C
C     ---- NOW CAN GET HEIGHT OF AXIS LABELS ----
C
      CHARH = AMIN1(XCHARH,YCHARH,TCHARH)
C
C     ---- NOW GET THE PAGE DEFINITION ----
C
      CALL PAGDEF(-(1.5+NYLNS*1.5+CHARH),-(1.5+(NXLNS+NTLNS)*1.5+CHARH),
     +   (1.0+XAXIS), (1.0+YAXIS) )
C
```

```
C  DO THE PLOTTING
C
C     ---- THE AXIS + SCALE MARKS ----
C
      CALL AXES(NXTICS,NYTICS,XSTEP,YSTEP,XSTART,YSTART,XVST,YVST)
C
C     ---- THE LABELS ----
C
      CALL LABEL(XLABEL,YLABEL,TITLE,NXCHAR,NYCHAR,NTCHAR,CHARH)
C
C     ---- THE CURVE ----
C
      CALL CURVE(X,Y,XSTART,YSTART,XFAC,YFAC,NPNTS)
C
C
      RETURN
      END
C
C*********************************************************************
C
      SUBROUTINE AXES(NXTICS,NYTICS,XSTEP,YSTEP,XSTART,YSTART,
     +    XVST,YVST)
C
C        THIS ROUTINE IS DESIGNED TO DRAW TO THE CURRENTLY ACTIVE
C     PLOT PAGE THE AXES & THE SCALE VALUES.
C
C
C
C        INPUTS
C        ======
C
C     NXTICS       NUMBER OF TICKS ON THE X AXIS
C
C     AYTICS
C     XSTEP
C
C     YSTEP
C
C     XSTART
C
C     YSTART
C
C     XVST
C
C     YVST
C
C        OUTPUT
C        ======
C
C
C        THIS ROUTINE WRITES TO THE CURRENTLY ACTIVE PLOT FILE THE CODE
C     NECCESSARY TO PRODUCE THE PLOT OF THE AXES & SCALES
C
C
C        DECLARATIONS
C        ============
C
C
      CHARACTER*4  FRMT(3) /'(  ','####','  )'/, CHAR(2)
C
      DATA  CHARH /0.35/
C
C
C     ---- SET UP THE SPACE SIZE OF THE TICMARK TO NUMBER GAP ----
C
      SPACE1 = CHARH
C
C     ---- SET UP THE TIC SIZE ----
C
      TICSIZ = 0.25*CHARH
C
C     ---- INITIALISE FOR X ----
C
      X = 0.0
      V = XSTART
      CALL FRMTIT(XSTEP,FRMT)
      DO 10 I = 1,NXTICS
        ENCODE(8,FRMT,CHAR)  V
        CALL PLTEXT((X-3.0*CHARH), -(SPACE1+TICSIZ+CHARH), CHARH, CHAR,
     +            0.0, 8)
C
C       ---- DRAW THE TIC MARK ----
C
        CALL DRWSEG(X, -TICSIZ, X, 0.0)
C
C       ---- SET UP FOR THE NEXT SHOT ----
C
        X = X + XSTEP
        V = V + XVST
C
10      CONTINUE
C
```

```
C
C     ---- DRAW THE AXIS LINE ----
C
      CALL DRWSEG(X-XSTEP,0.0,0.0,0.0)
C
C     ---- NOW THE Y AXIS ----
C
C
C     ---- INITIALISE ----
C
      Y = 0.0
      V = YSTART
      CALL FRMTIT(YSTEP,FRMT)
      DO 20 I = 1,NYTICS
        ENCODE(8,FRMT,CHAR)  V
        CALL DTEXT(-(SPACE1+TICSIZ), (Y-5.0*CHARH), CHARH, CHAR,
     +       90.0, 8)
C
C       ---- DRAW THE TIC MARKS ----
C
        CALL DRWSEG(-TICSIZ, Y,  0.0, Y)
C
C       ---- SET UP FOR THE NEXT SHOT ----
C
        Y = Y + YSTEP
        V = V + YVST
C
20    CONTINUE
C     ---- DRAW THE AXIS LINE ----
C
      CALL DRWSEG(0.0,Y-YSTEP,0.0,0.0)
C
C
C  THATS IT
C
      RETURN
      END
C
C*************************************************************************
C
      SUBROUTINE  FRMTIT(STEP, FRMT)
C
C        THIS ROUTINE IS DESIGNED TO GENERATE THE FORMAT CODE
C     TO CONVERT THE VALUES SEPERATED BY STEP INTO CHARACTERS.
C
C        INPUT
C        =====
C     STEP         THE CHANGE IN VALUE BETWEEN TWO SUCCESSIVE VALUES
C                   TO BE ENCODED
C
C        OUTPUT
C        ======
C
C     FRMT          THE CHARACTER ARRAY GIVING THE FORMAT
C
C
C
C        DECLARATIONS
C        ============
C
C
      CHARACTER*4  FRMT(1)
C
C
      IF( STEP .GE.  10000.00 .OR.  STEP .LT.  -1000.00)  THEN
C
C       ---- NUMBER WILL HAVE TO BE IN E FORMAT ----
C
        FRMT(2) = 'E8.3'
C
C
      ELSE IF(ABS( STEP) .GT. 1.00) THEN
C
C       ---- NUMBER MAY BE IN F FORMAT ----
C
        FRMT(2) = 'F8.2'
C
      ELSE IF(ABS( STEP) .GT. 0.001 ) THEN
C
C       ---- WILL BE FORCED TO USE E FORMAT ----
C
        FRMT(2) = 'E8.3'
C
      ELSE
C
C       ---- CAN USE MODIFIED F FORMAT ----
C
        FRMT(2) = 'F8.5'
C     END IF
C     RETURN
      END
```

```
C
C*************************************************************************
C
      SUBROUTINE  LABEL(XLABEL,YLABEL,TITLE,NXCHAR,NYCHAR,NTCHAR,CHARH)
C
C        THIS ROUTINE WRITES THE X & Y LABELS AND TITLE
C
C
C        INPUTS
C        ======
C
C     XLABEL       THE CHARACTER ARRAY CONTAINING THE X LABEL
C
C     YLABEL       THE CHARACTER ARRAY CONTAINING THE Y LABEL
C
C     TITLE        THE CHARACTER ARRAY CONTAINING THE PLOT TITLE
C
C     NXCHAR       THE NUMBER OF CHARACTERS IN THE X LABEL
C
C     NYCHAR       THE NUMBER OF CHARACTERS IN THE Y LABEL
C
C     NTCHAR       THE NUMBER OF CHARACTERS IN THE PLOT TITLE
C
C     CHARH        THE HEIGHT TO DRAW THE CHARACTERS
C
C
C        OUTPUT
C        ======
C
C        TO THE CURRENTLY ACTIVE GDP PLOT PAGE THE LABELS ARE DRAWN
C
C
C        DECLARATIONS
C        ============
C
C
      CHARACTER*4  XLABEL(1), YLABEL(1), TITLE(1)
C
C
C
C  DO THE X AXIS LABEL
C
C
      Y = -(1.2 + CHARH)
C
        CALL DTEXT(0.0, Y, CHARH, XLABEL, 0.0, NXCHAR)
C
C  NOW THE TITLE
C
      Y = -(1.2 + 2.5*CHARH)
C
      CALL DTEXT(0.0, Y, CHARH, TITLE, 0.0, NTCHAR)
C
C
C  AND THE Y AXIS LABEL
C
      X = -1.2
C
        CALL DTEXT(X, 0.0, CHARH, YLABEL, 90.0, NYCHAR)
C
      RETURN
      END
C
C*************************************************************************
C
      SUBROUTINE  CURVE(X, Y, XSTART, YSTART, XFAC, YFAC, NPNTS)
C
C        THIS ROUTINE DRAWS A LINE GRAPH OF THE X PTS VS THE
C     Y POINTS
C
C        INPUTS
C        ======
C
C     X            ARRAY OF X POINTS
C
C     Y            ARRAY OF Y POINTS
C
C     XSTART       STARTING VALUE OF X ARRAY
C
C     YSTART       STARTING VALUES OF Y ARRAY
C
C     XFAC         FACTOR TO CONVERT ABSOLUTE X VALUES TO CMS
C
C     YFAC         FACTOR TO CONVERT ABSOLUTE Y VALUES TO CMS
C
C     NPNTS        THE NUMBER OF POINTS IN THE X & Y ARRAYS
C                  TO BE PLOTTED
C
```

```
C
C       DECLARATIONS
C       ============
C
C
      REAL  X(1), Y(1)
C
C
C
C  GET THE SHOW ON THE ROAD
C
      CALL MOVE((X(1)-XSTART)*XFAC,(Y(1)-YSTART)*YFAC)
C
      DO 10 I = 2,NPNTS
C
        CALL DRAW((X(I)-XSTART)*XFAC,(Y(I)-YSTART)*YFAC)
C
10    CONTINUE
C
C
      RETURN
      END
END OF FILE
->
```

A PROOF ON THE CONVOLUTION RESULTS USED

The proof given here shows that when relating the input x(t) to a system with impulse response g(t) and output y(t) via the convolution integral the exact relation holds if one substitutes the autocorrelation function of the input for the input x(t) and the crosscorrelation function of the input and output for the output y(t).

Defining the autocorrelation of a function x(t) as:

$$\phi_{xx}(t) = \int_{-\infty}^{\infty} x^*(t).x(t+\tau)dt \qquad \ldots A4.1$$

and the crosscorrelation of two functions x(t) and y(t) as:

$$\phi_{xy}(\tau) = \int_{-\infty}^{\infty} x^*(t).y(t+\tau)dt \qquad \ldots A4.2$$

If we then state the convolution integral relating x(t), g(t) and y(t):

$$y(t) = \int_{0}^{\infty} g(\theta).x(t-\theta)d\theta \qquad \ldots A4.3$$

Now replacing x(t) with its autocorrelation function $\phi_{xx}(\tau)$ and y(t) by some unknown function I, we may then write:

$$I = \int_0^\infty g(\theta).\emptyset_{xx}(\tau-\theta)d\theta$$

Now substituting for $\emptyset_{xx}(\tau-\theta)$ from equation A4.1 we have:

$$I = \int_0^\infty g(\theta) \int_{-\infty}^\infty x^*(t).x(t+\tau-\theta)dt\ d\theta$$

rearranging the order of integration

$$I = \int_0^\infty\int_{-\infty}^\infty g(\theta)\ x^*(t).x(t+\tau-\theta)dt\ d\theta$$

$$= \int_{-\infty}^\infty x^*(t) \int_0^\infty g(\theta).x(t+\tau-\theta)d\theta\ dt \qquad \ldots \text{ A4.5}$$

and since we recognise that:

$$\int_0^\infty g(\theta)x(t+\tau-\theta)d\theta = g(t)*x(t+\tau) = y(t+\tau) \quad \text{by A4.3}$$

Substituting this into A4.5 we have that:

$$I = \int_{-\infty}^\infty x^*(t)y(t+\tau)dt = \emptyset_{xy}(\tau) \qquad \text{by A4.2}$$

thus we may rewrite expression A4.3:

$$\emptyset_{xy}(\tau) = \int_0^\infty g(\theta).\emptyset_{xx}(\tau-\theta)d\theta$$

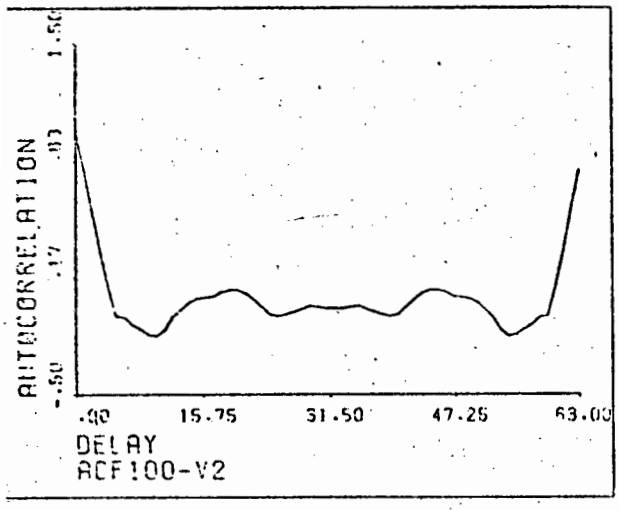


Figure A5.1a The autocorrelation function.

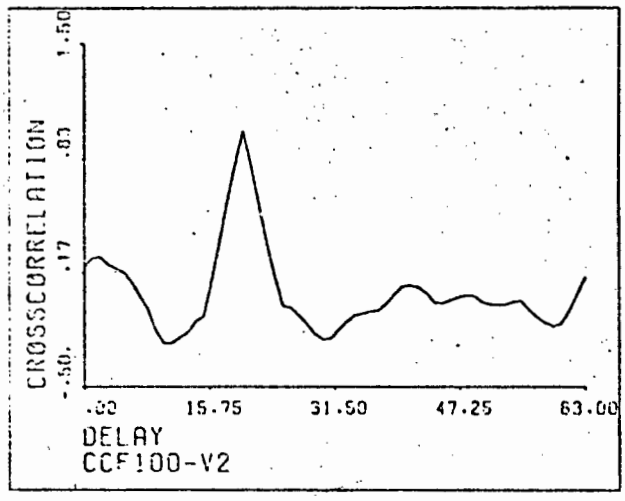


Figure A5.1b The crosscorrelation function.

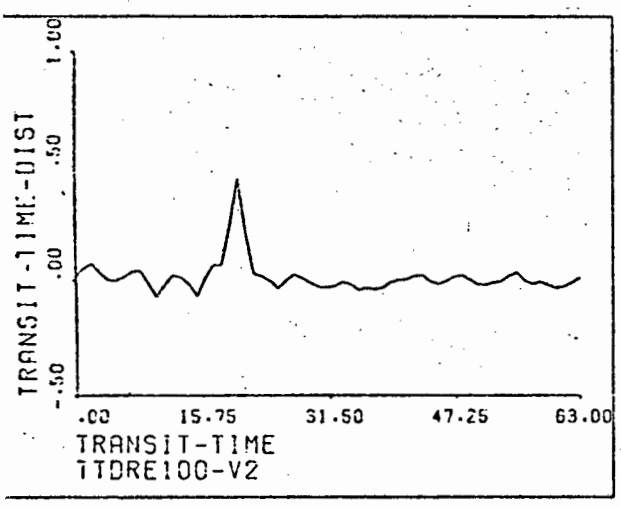


Figure A5.1c The transit time distribution.

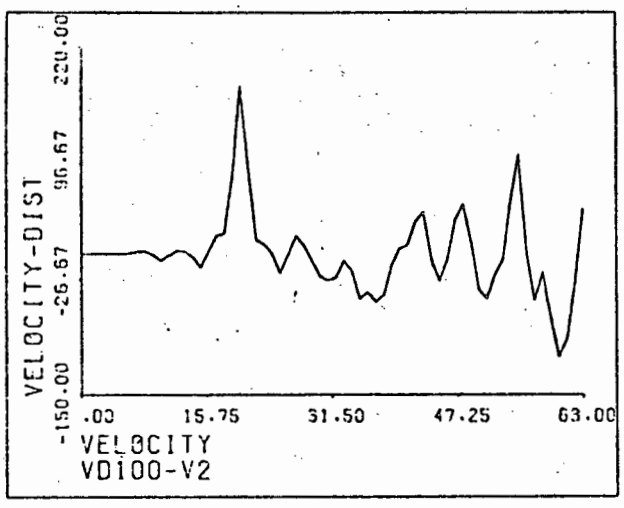


Figure A5.1d The velocity distribution.

```
SIMULA        V2.0        17/02/81


PIPE LENGTH =  50      PIPE WIDTH =  20

NUMBER OF SAMPLES =   64      AVERAGING WIDTH =  10


NUMBER OF RECORDS AVERAGED =     100

VELOCITY PROFILE USED IS :

   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2

ACF NORMALIZING FACTOR =        7.7483      CCF NORMALIZING FACTOR =        5.1628
```

Figure A5.1e The simulation program output using a flat velocity profile.

APPENDIX 5

DECONVOLUTION OF CORRELATION FUNCTIONS OBTAINED FROM THE SIMULATED FLOW SYSTEM

The flow simulation program (SIMULA) was run using stepped and flat velocity profiles. The autocorrelation and crosscorrelation functions thus calculated were then deconvolved in an effort to arrive back with a transit time distribution or velocity distribution which resembles the original distribution used in the simulation of these correlation functions.

The output from the simulation program, the respective correlation functions and deconvolution results are shown here for three different velocity profiles. In Figure A5.1 a flat velocity profile was used in the simulation of the flow and hence the TTD and VD are expected to comprise a single impulse corresponding to this delay. If one examines Figure A5.1c and A5.1d, these impulses are present. In Figure 5.1d however, the velocity distribution has a poor signal to noise ratio. This may be attributed to the finite averaging effects as discussed in Section 3.1.4.

The curves shown in Figure A5.2 were also obtained from a simulation with a flat velocity profile but in this case the flow velocity has been increased. The ACF is narrower than that of Figure A5.1 due to the increase in frequency

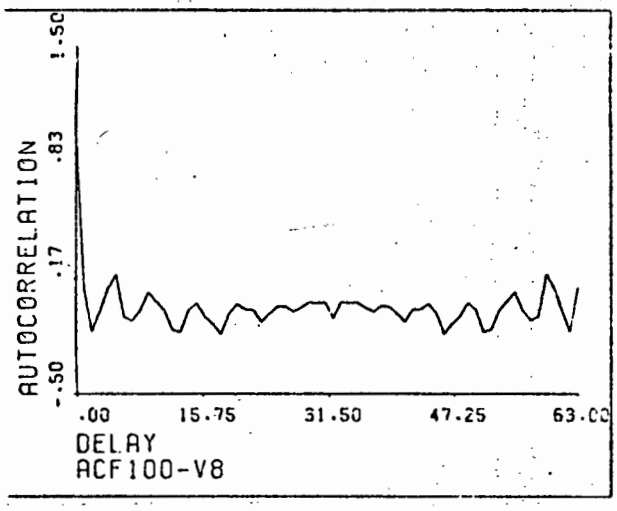


Figure A5.2a The autocorrelation function.

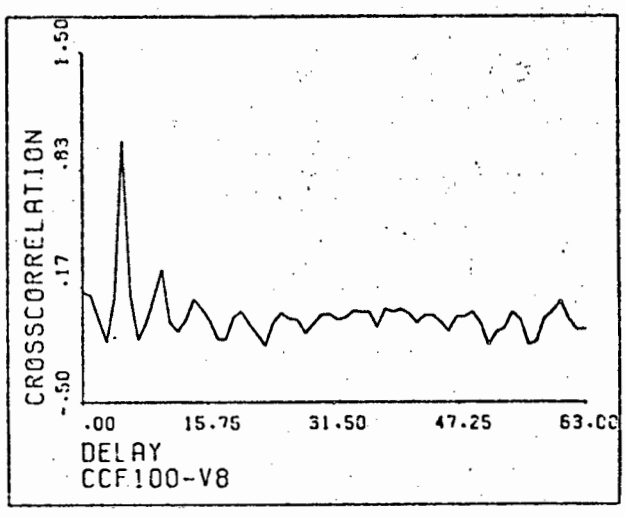


Figure A5.2b The crosscorrelation function.

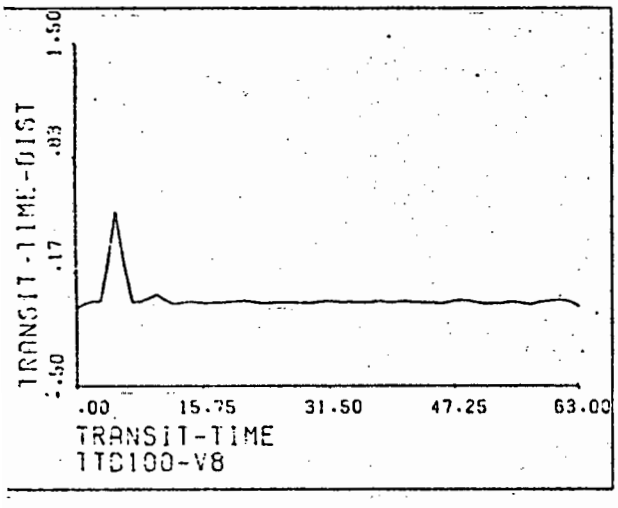


Figure A5.2c The transit time distribution.

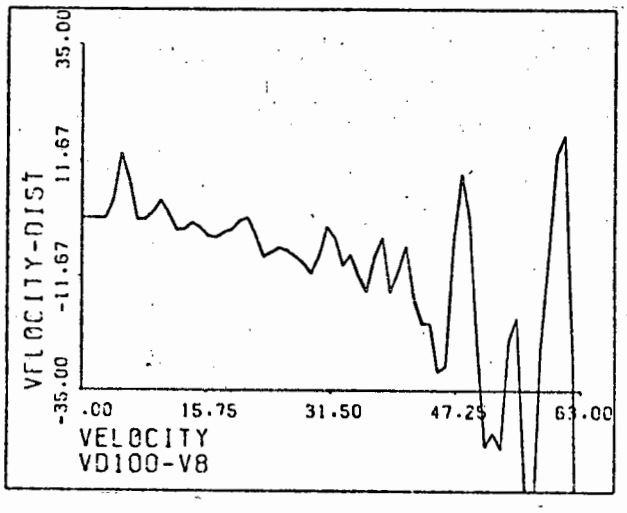


Figure A5.2d The velocity distribution.

```
SIMULA        V2.0        17/02/81

PIPE LENGTH =  50      PIPE WIDTH =  20

NUMBER OF SAMPLES =   64      AVERAGING WIDTH =  10

NUMBER OF RECORDS AVERAGED =     100

VELOCITY PROFILE USED IS :

   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8

ACF NORMALIZING FACTOR =        7.9484      CCF NORMALIZING FACTOR =        7.3592
```

Figure A5.2e The simulation program output using a flat velocity profile.

of the flow signals arising from the more rapid flow. The resulting TTD and VD are shown in Figures A5.2c and A5.2d. In Figure A5.3 the deconvolution of correlation funtion simulated using a stepped velocity profile are given.

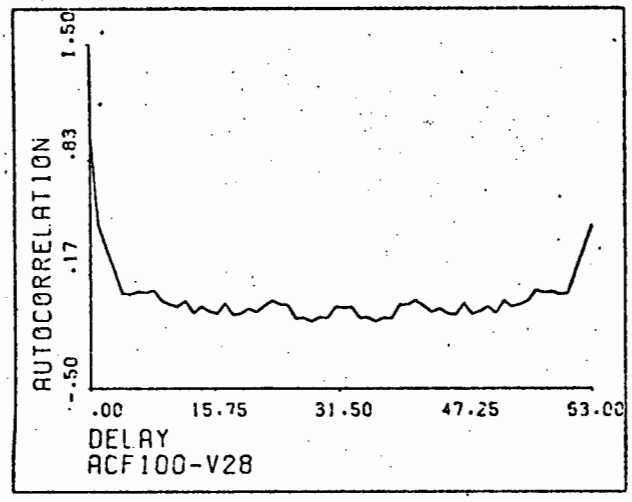


Figure A5.3a The autocorrelation function.

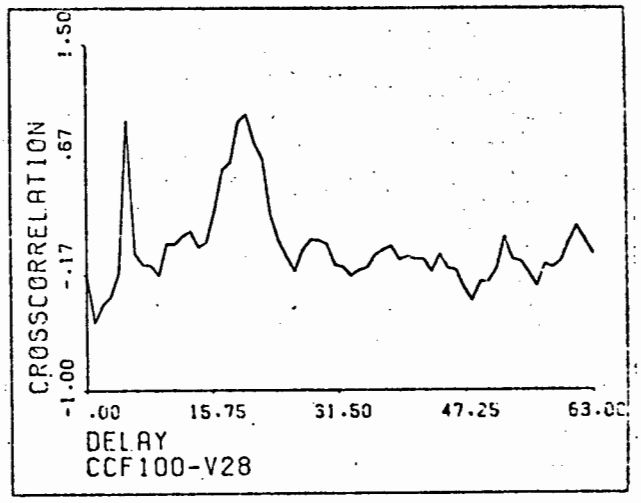


Figure A5.3b The crosscorrelation function.

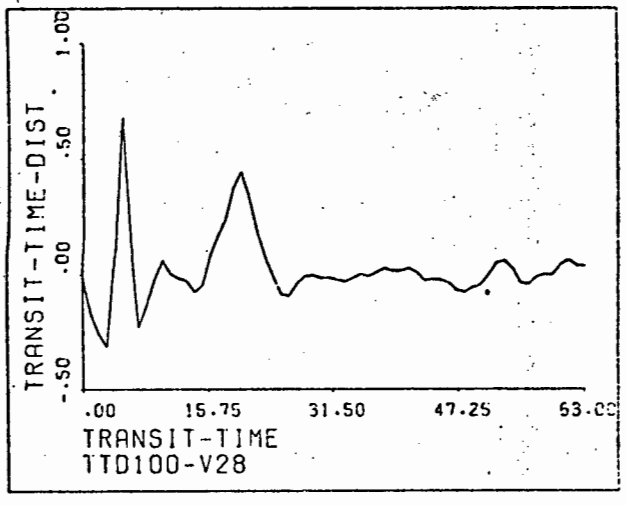


Figure A5.3c The transit time distribution.

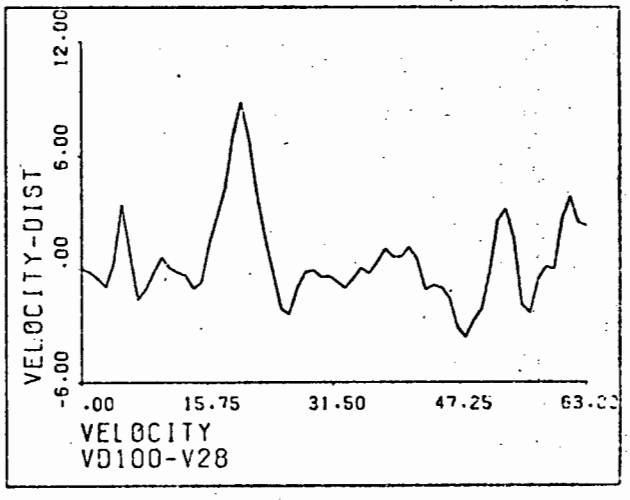


Figure A5.3d The velocity distribution.

```
SIMULA        V2.0        17/02/81

PIPE LENGTH =  50      PIPE WIDTH =  20

NUMBER OF SAMPLES =   64      AVERAGING WIDTH =  10

NUMBER OF RECORDS AVERAGED =     100

VELOCITY PROFILE USED IS :

   2   2   2   2   2   2   2   2   2   2   8   8   8   8   8   8   8   8   8   8

ACF NORMALIZING FACTOR =        8.3858      CCF NORMALIZING FACTOR =        3.1288
```

Figure A5.3e The simulation program output using a stepped velocity profile.

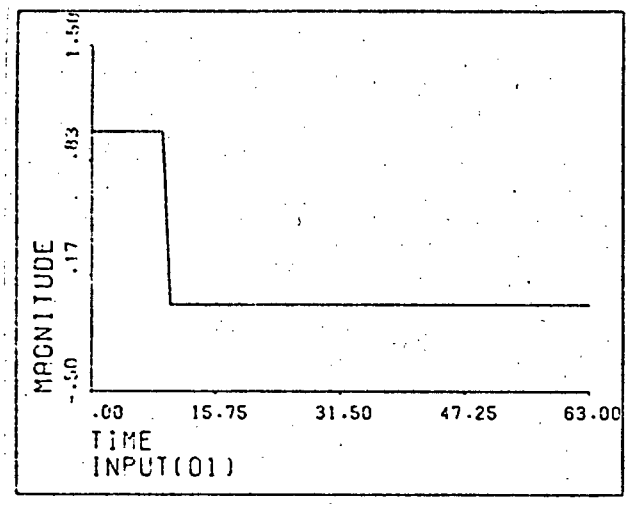


Figure A6.1a Rectangular input function.

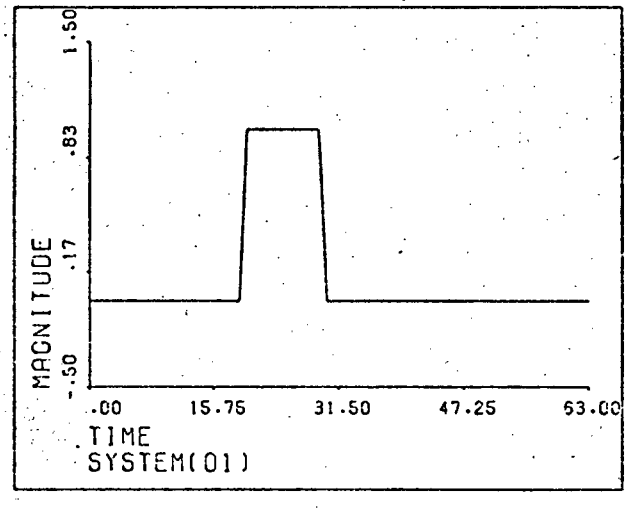


Figure A6.1b Rectangular system impulse response.

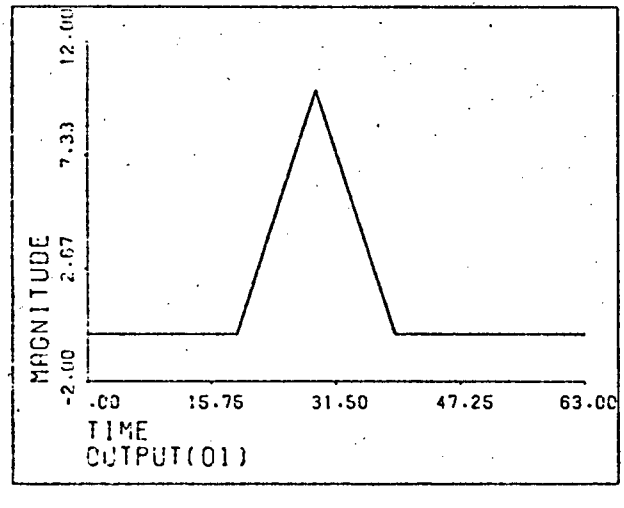


Figure A6.1c Triangular system output.

APPENDIX 6

FOURIER TRANSFORM DECONVOLUTION IN THE PRESENCE OF NOISE

In an effort to determine the sensitivity of the Fourier transform deconvolution method to noise, several tests were carried out. Two well-known functions were convolved and this result stored. Noise signals of different amplitudes are then added to this convolution result and the deconvolution of the original function and this noisy function carried out. Results of the deconvolution of triangular and rectangular pulses are shown here as well as those obtained using "test" correlation functions. These "test" correlation functions were computed using the autocorrelation function measured from the flow rig. The crosscorrelation function was found by computing a transit time distribution resulting from a typical parabolic velocity profile and convolving this TTD with the ACF.

The rectangular functions used in the first tests are shown in Figures A6.1a and A6.1b. The convolution of these two pulses is given in Figure A6.1c. Various levels of noise are added to this triangular waveform and the deconvolution of the rectangular function (in Figure A6.1a) and this noisy triangular function carried out. The expected result is the waveform given in Figure A6.1b.

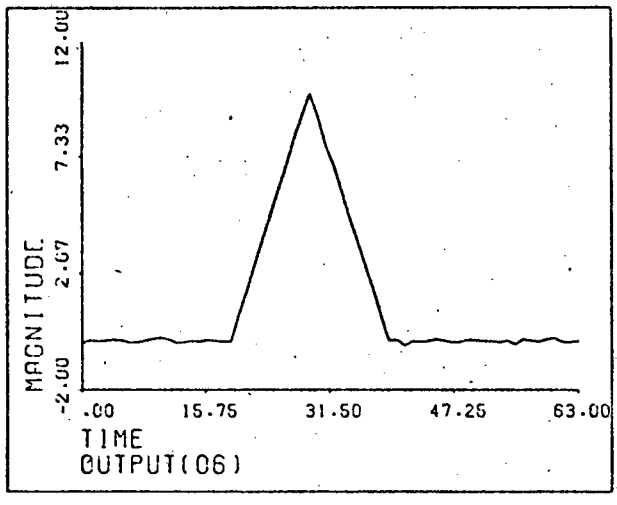


Figure A6.2a System output + noise..

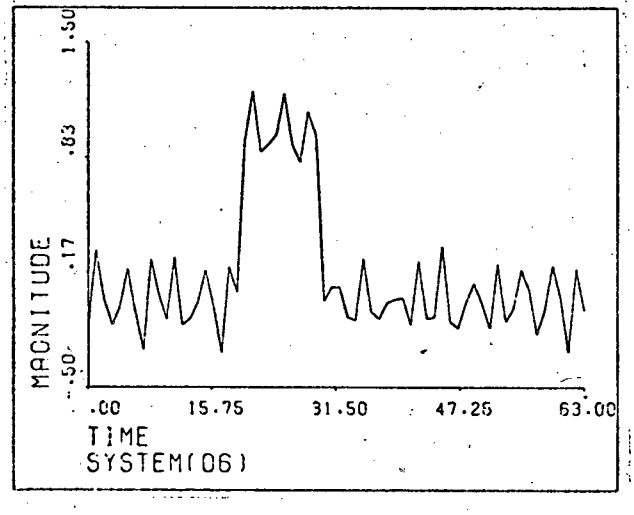


Figure A6.2b Deconvolved impulse response without windowing.

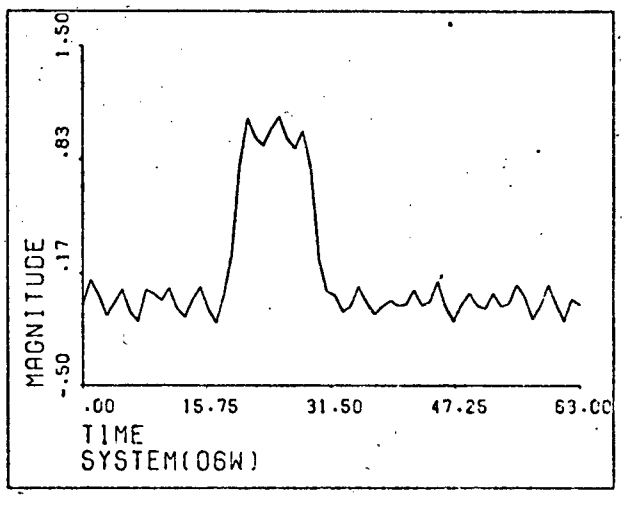


Figure A6.2c Deconvolved impulse response using Hamming windowing.

In Figures A6.2a and A6.3a the noise signals are added to the triangular function. The deconvolution results shown in Figures A6.2b and A6.3b have been computed without applying any windowing to the intermediate deconvolution results. The results shown in Figures A6.2c and A6.3c have been calculated by applying Hamming windows to the Fourier transform of the autocorrelation and crosscorrelation functions before and after dividing these Fourier transforms.

The application of the data window definitely reduces the noise present in the deconvolved function. The results demonstrate the sensitivity of the deconvolution result to noise added to the waveform to be deconvolved.

A second test was carried out where the functions represent more closely typical autocorrelation and crosscorrelation functions. Examples are given in Figures A6.4a and A6.4b. The TTD used to compute the CCF is given in Figure A6.4c. The deconvolution is thus expected to return a TTD similar to that shown in Figure A6.4c.

With no added noise, the deconvolution returns the expected TTD. Extremely low noise signals which were added to the CCF are shown in Figures A6.5b, A6.6b and A6.7b for comparison with the original CCFs. The correlation functions with the noise signals added, are shown in Figures A6.5a and A6.6a and A6.7a. Hamming windowing was

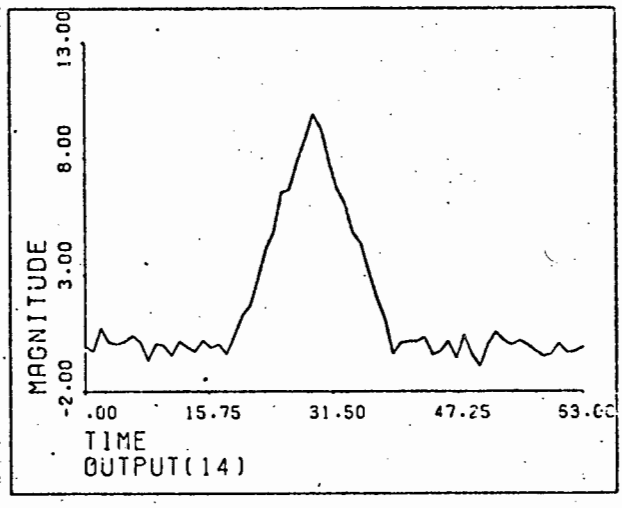


Figure A6.3a System output + noise.

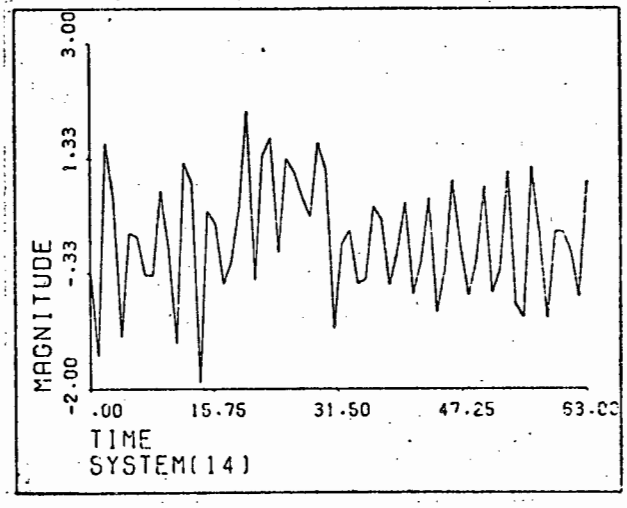


Figure A6.3b Deconvolved impulse response without windowing.

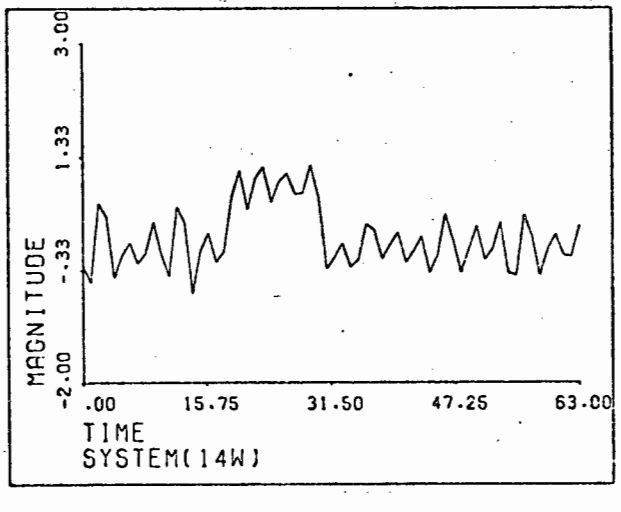


Figure A6.3c Deconvolved impulse response using Hamming windowing.

applied to the intermediate deconvolution results. The TTDs obtained by deconvolving these correlation functions are shown in Figures A6.5c, A6.6c and A6.7c.

These results show the typical correlation functions to be much more sensitive than the rectangular functions to added noise when deconvolving.

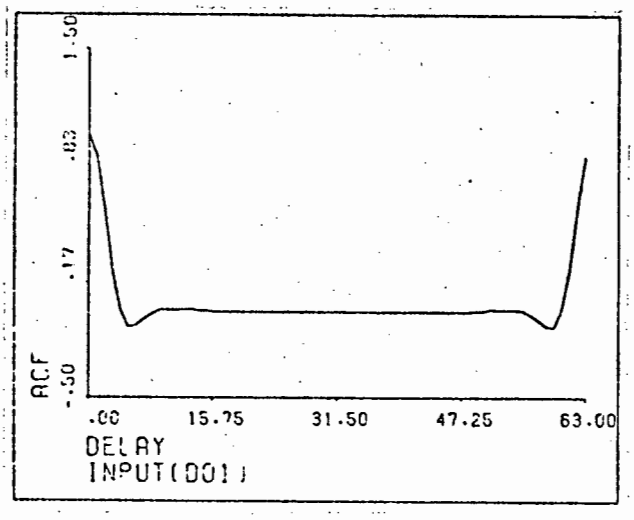


Figure A6.4a The ACF measured from the flow rig.

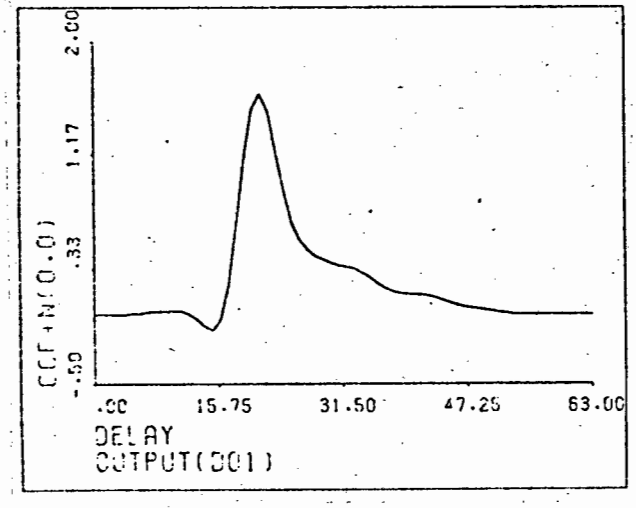


Figure A6.4b The CCF obtained from the convolution of the ACF and the TTD.

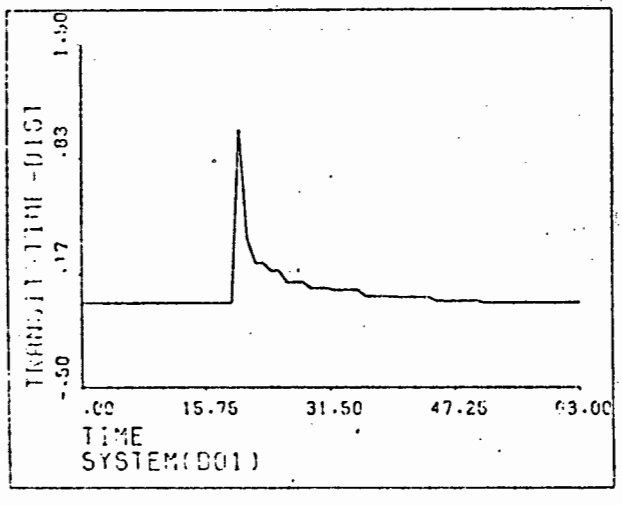


Figure A6.4c The TTD calculated from a parabolic velocity profile.

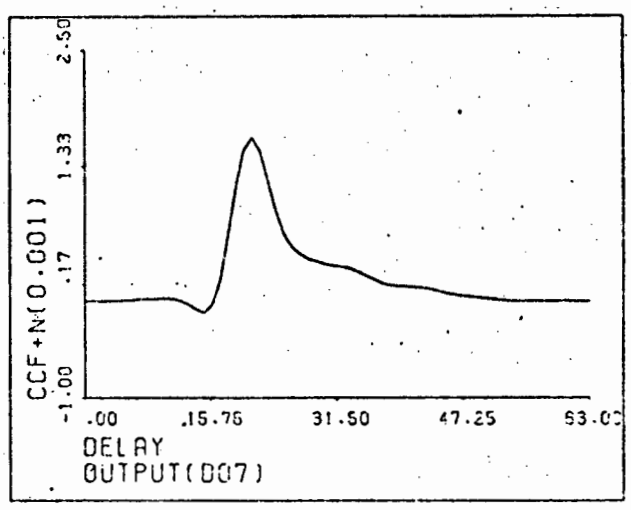


Figure A6.5a CCF + noise (.001)

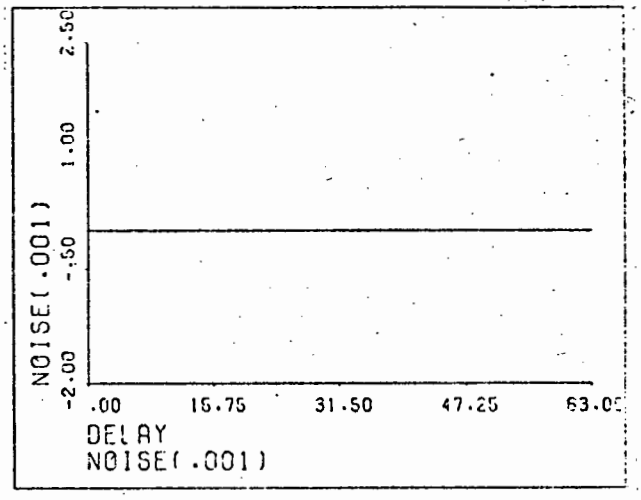


Figure A6.5b The noise added to the CCF

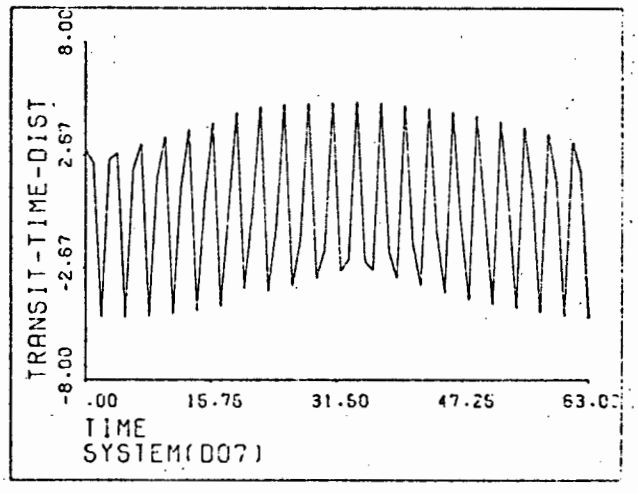


Figure A6.5c The TTD obtained from the deconvolution of the ACF and the CCF + noise (.001)

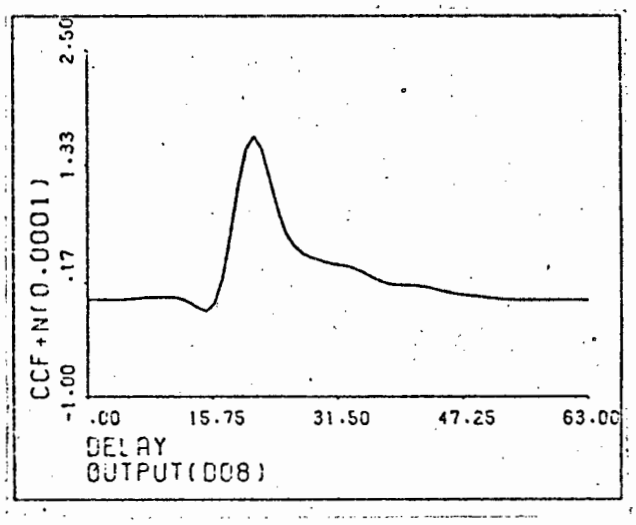


Figure A6.6a CCF + noise (.0001)

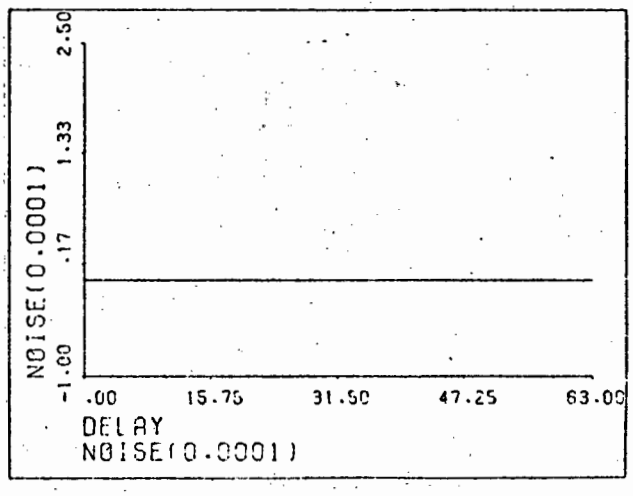


Figure A6.6b The noise added to the CCF

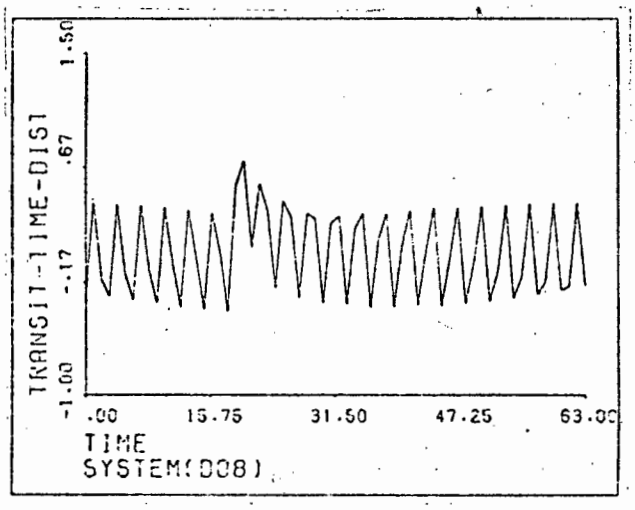


Figure A6.6c The TTD obtained from the deconvolution of the ACF and the CCF + noise (.0001)

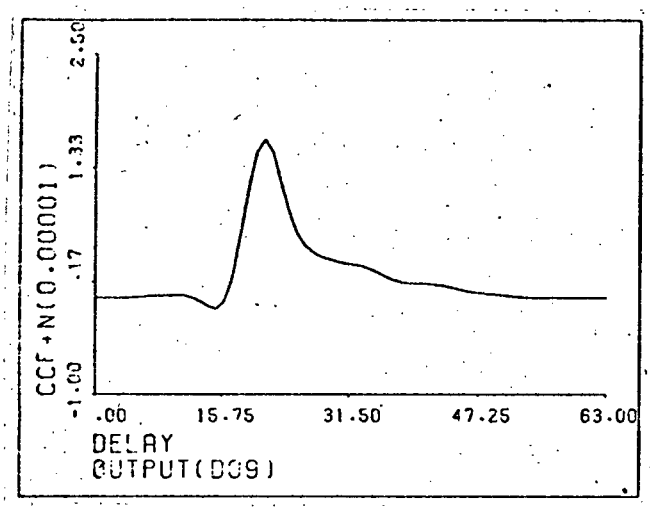


Figure A6.7a The CCF + noise (.00001)

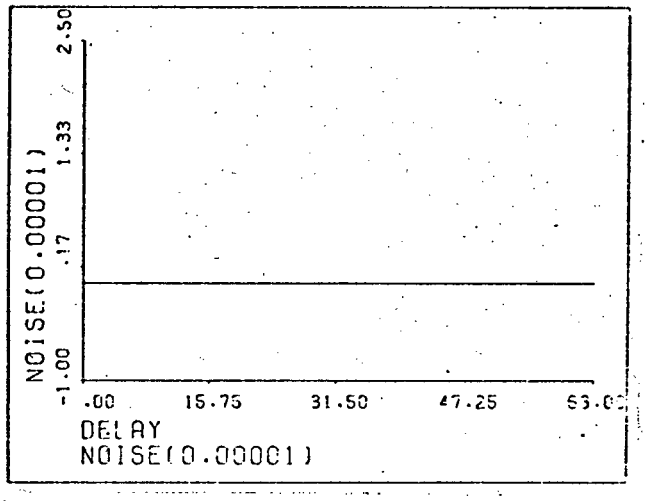


Figure A6.7b The noise added to the CCF

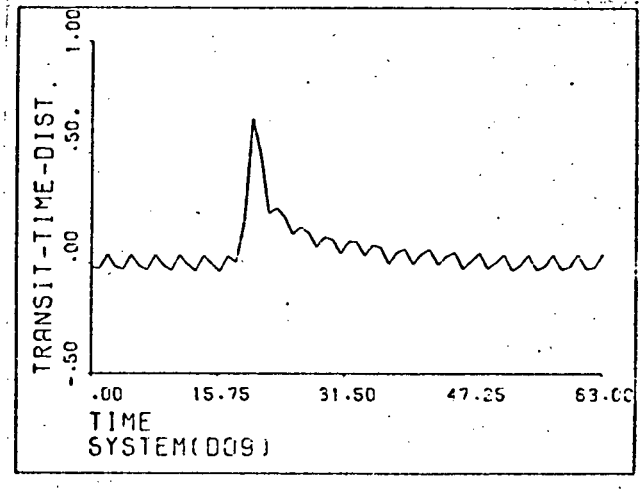


Figure A6.7c The TTD obtained from the deconvolution of the ACF and the CCF + noise (.00001)